UNIS
The University Centre in Svalbard

ENSTA BRETAGNE

UNIVERSITY CENTRE IN SVALBARD

MASTER'S THESIS

Automatic morphological classification of auroral structures

Author:
Vincent TEISSIER

Supervisor:
Pr. Noora PARTAMIES

FISE 2022
Observation Systems and Artificial Intelligence (SOIA)
ENSTA BRETAGNE

NOT CONFIDENTIAL

August 20, 2022

Abstract

Vincent TEISSIER

Automatic morphological classification of auroral structures

English

In this project we wanted to automatically find groupings in auroral all-sky color images from the Kjell Henriksen Observatory in Svalbard, according to the morphology of the auroras, and this without any manual labelling. The data set used contained only clear skies with auroras and sometimes with the Moon. We used 2 feature extractors based on unsupervised deep-learning (Convolutional Autoencoder, SimCLR network) and 3 Convolutional neural networks pretrained on the ImageNet dataset (Resnet-50, Inception-v3, MobileNet-v2). We used 4 different dimension reduction methods on the feature data (PCA, Kernel PCA, Isomap and UMAP), followed by 3 simple clustering methods (K-means, Spectral and Hierarchical clustering) and a constrained clustering method (COP K-means). All computed combinations of the listed methods were ranked according to internal validation indices and validated by manual visual inspection of the data. For each feature extractor, the best results were combined using a majority voting principle in order to provide more complex groupings of the data. The fusion result which originates from a SimCLR feature extractor gave the most relevant partition of the data, by grouping together images containing similar auroral features such as vortex-like structures or patchy texture. However the presence of the Moon in the image constituted an important bias in the final feature data representation.

Français

L'objectif de ce projet est de trouver automatiquement des groupements au sein d'un jeu de données d'images couleur de l'ensemble du ciel contenant des aurores boréales. Les groupements en question doivent correspondre à des types de morphologies d'aurores, et ce groupement doit se faire avec le moins de biais possible de la part de l'Homme. Le jeu de données provient de l'observatoire Kjell Henriksen situé dans l'archipel du Svalbard, et a été préparé pour ne contenir que des images de ciel nocturne sans nuage avec une activité aurorale, avec ou sans la Lune. On a utilisé 2 extracteurs de caractéristiques basés sur de l'apprentissage profond non-supervisé (Autoencodeur Convolutif et réseau SimCLR) et 3 réseaux de neurones convolutifs préentrainés sur le jeu de données ImageNet (Resnet-50, Inception-v3, MobileNet-v2). On a appliqué ensuite 4 méthodes de réduction de dimension sur les caractéristiques extraites (PCA, Kernel PCA, Isomap et UMAP), suivies de 3 méthodes de clustering simple (clustering K-means, Spectral et Hierarchique), et d'une méthode de clustering contraint (COP K-means). Toutes les combinaisons des méthodes citées précédemment ont ensuite été classées selon des scores de validation interne et validées par inspection visuelle des résultats. Pour chaque extracteur de caractéristiques, les meilleurs résultats ont été fusionnés en utiliant un principe de vote à majorité, afin de produire des groupements plus complexes. Le résultat de fusion provenant de l'extracteur de caractéristiques SimCLR a fourni le groupement le plus pertinent, en groupant entre eux des images contenant des caractéristiques aurorales similaires, comme par exemple des aurores brillantes en forme de tourbillon ou des aurores présentant une texture plus irrégulière. Cependant la présence de la Lune sur une partie des images constitue un biais important dans la construction de la représentation finale des données.

Acknowledgements

I would like to thank my supervisor, Pr. Noora Partamies for helping me throughout the project as well as with the technical difficulties of living in Svalbard.

I would also like to thank Mikko Syrjäsuo for his help and comments on the technical details of the project.

Thank you also to Abdelmalek Toumi for providing support from the administrative point of view as well as for giving ideas for beginning the project.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

Longyearbyen is a town located in the Svalbard archipelago, which is around 78°N in latitude, north from mainland Norway. Because of Earth's inclination, during the winter the sun does not go above the horizon. As a result, during around 3 months, there is a continuous night. In addition, above the polar circle take place specific phenomena caused by the interaction between the Sun's and the Earth's magnetic fields: auroras. Because they can only be observed during the night, a location such as Longyearbyen is ideal for observing and studying auroras.

In this town is located the University Centre in Svalbard (UNIS), which is the northernmost education and research centre founded in 1993 focused on Arctic Geology, Geophysics, Biology and Technology. UNIS has set up in 2007 an observatory near the town called the Kjell Henriksen Observatory (KHO)[1]. This observatory hosts a wide range of instruments for scientific projects worldwide. One of them which is operated by UNIS is an optical color camera (Sony a7s) (see Figure 1.1). This instrument is capable of imaging the whole sky thanks to a fish-eye lens. During the night, it takes one picture every 12 seconds in order to capture auroral lights. As it is operating 24 hours a day during the 3 months of winter, it creates a large amount of data which needs to be processed in order to study the occurrences and the structures of auroras.

FIGURE 1.1: View of the roof of the Kjell Henriksen Observatory (left) and of the dome under which is located the camera used in this project (right)

1.2 Problematic

One of the challenges of studying auroras is to categorize auroral structures into well-defined categories. The structure of the aurora can be defined in terms of shape, contrast, color or even evolution over time. They can be defined either perceptually or depending on the physical phenomenon from which an aurora originates. For example, the most

common auroral structure is the arc-like shape with a bright green color. Others are fainter and take up the whole sky, and some have distinct red and violet colors.

Another issue with characterizing auroras is that they are not solid objects, thus the light emission is transparent. In most cases, we can see the night sky through the auroral light. As a result, changes in the background can influence the appearance of an aurora. This can happen when the moon appears in the sky, when the sun is not far below the horizon, or when light pollution occurs. The presence of clouds in front of an aurora can also change its luminosity and shape (see Figure 1.2).

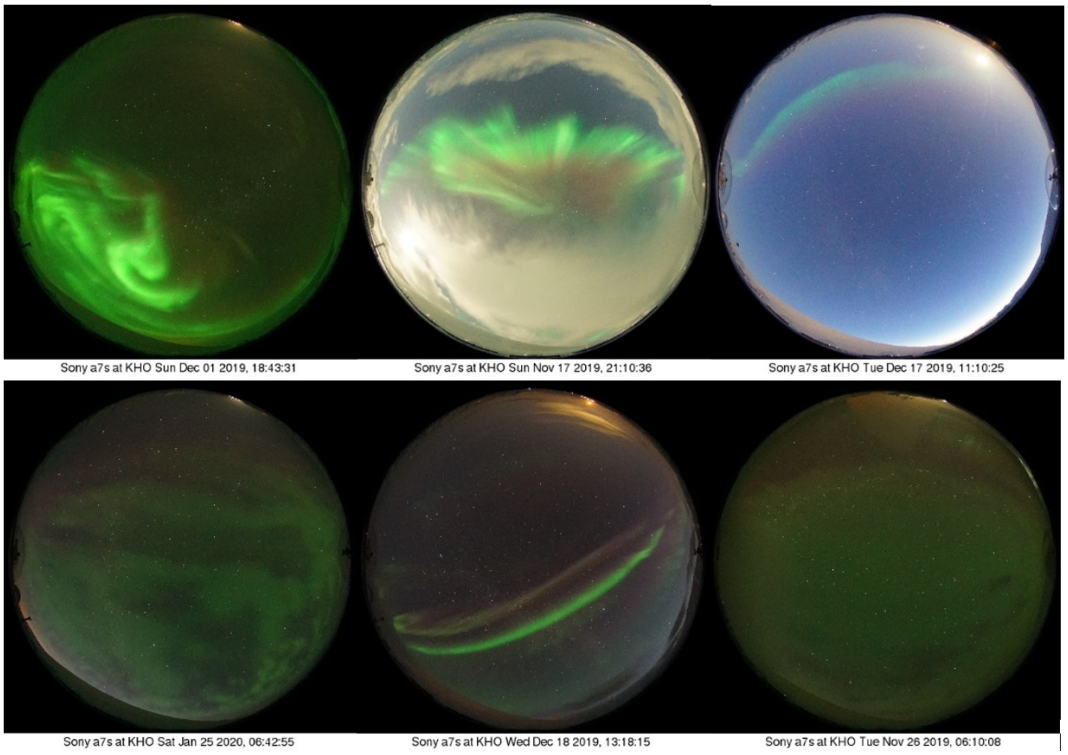


FIGURE 1.2: Examples of auroral images available in our data set, showing the diversity of auroral forms and sky conditions

1.3 Goal

The goal of this project is to automatically classify auroral structures into groups which are as distinct as possible, and which can ideally represent perceptual characteristics or physical phenomena. The challenge here is to avoid as much as possible the human biases for the choice of auroral types. This is why we will use unlabelled images, in order to perform unsupervised classification of auroral structures solely based on the image data. This will hopefully allow to statistically study the occurrences of different types of auroral morphology and better correlate them with the physical conditions of the Earth's atmosphere and the near-Earth space.

2 Auroras

Auroras are geophysical phenomena which occur in the atmosphere (from 100 to 400 km high) and can be commonly observed at high north and south latitudes. They appear as a bright and transparent light emission, mainly with green color, but sometimes also blue and red, and come with a wide diversity of shapes, brightness, and textures.

These auroras are the consequence of the precipitation of charged particles coming from the Sun which are captured by the magnetosphere, the impact area of the Earth's magnetic field [2].

2.1 Sources of emission from the Sun

There are three types of solar particle emission: Coronal Mass Ejections, Solar flares and Solar wind.

Coronal Mass Ejections or CMEs are caused by realignments of the Sun's magnetic fields. During these events, segments of the outer corona are expelled from the Sun in the form of a plasma cloud. Such plasma clouds contain mainly free electrons, but also protons and other ions, which come in majority from atoms of Hydrogen and Helium present in the Sun's outer layers.

CMEs are not very energetic; however they can create a shock wave in the solar wind flow that accelerates them, giving them more energy and more chance to precipitate into the Earth's atmosphere.

Solar flares, like CMEs, consist in a conversion of magnetic energy resulting in an acceleration of charged particles as a plasma. However they are much more localized, into magnetically unstable regions of the Sun. In addition, a part of the energy is released as a short and intense burst of X-ray and Extreme Ultraviolet electromagnetic waves.

Solar wind consists in slow or fast streams of charged low-energy particles. Unlike solar flares or CMEs, it appears as a continuous flow of plasma. It can however be accelerated by fast CMEs and be more energetic than solar flares. Solar wind particles are ejected along open magnetic field lines, and have a shape of individual stream lines. Fast Solar wind particles come from coronal holes, which are regions of lower density, without concentrated magnetic fields. These are mainly localized at the Sun's poles during phases of low solar activity but can go to lower latitudes during a maximum of solar activity.

2.2 Interactions with the Earth

When the solar wind approaches the Earth, it interacts with the Earth's magnetic field, called the magnetosphere (see Figure 2.1). The outer boundary of this magnetosphere is called the magnetopause and is located around 64000 km above the Earth's surface. Because of the constant solar wind, the magnetosphere is compressed on the side facing the Sun and extended outwards in the night side, in a region called the magnetotail. The

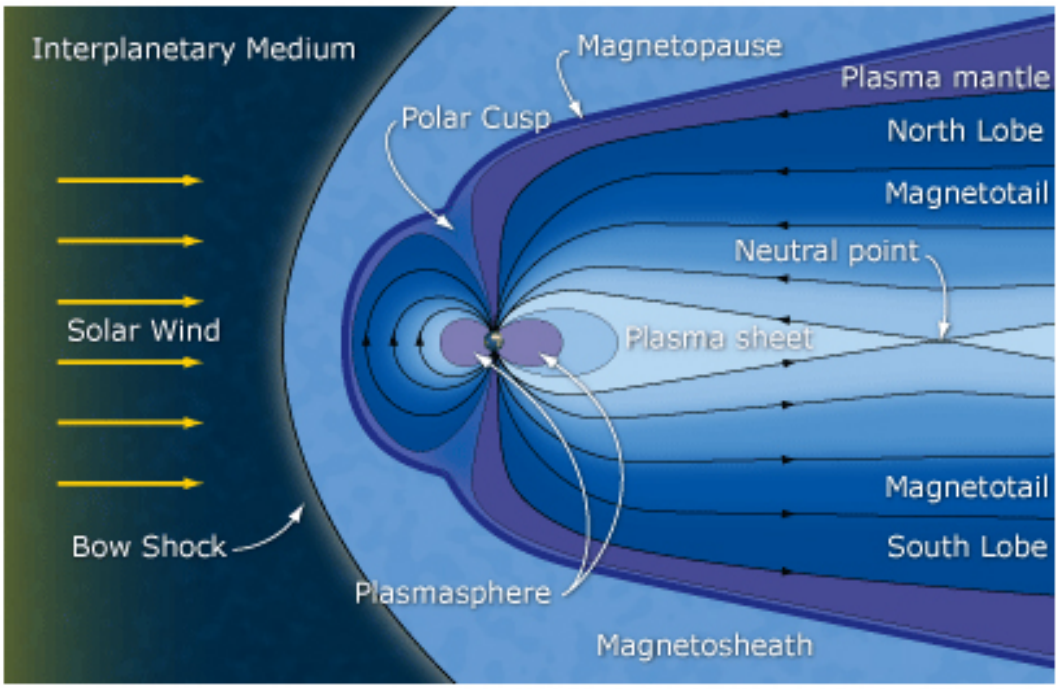


FIGURE 2.1: Diagram of the magnetic near-Earth environment. [2]

Earth's magnetosphere surrounded by solar wind plasma acts in almost the same way as a solid object in a moving fluid. Indeed, 16000 km ahead of the magnetopause appears a stationary shock wave which is called the bow shock and is 100 km wide. This bow shock reduces the speed of the incoming solar wind and creates turbulence in the particle flow. When arriving at the magnetopause, a majority of particles are deflected along the border. However, in some cases, solar particles can enter the magnetosphere and precipitate into the atmosphere.

One of the most important parameter influencing particles entry into the magnetosphere is the magnetic polarisation of the solar wind. If its vertical component has the same North-South orientation as the Earth, the Sun's and the Earth's magnetic fields repel each other. Thus, few solar particles enter the magnetosphere, except in the regions near the poles called the polar cusps, where the magnetic field is weaker, allowing a portion of the solar wind to access the atmosphere. Another entry point into the magnetosphere is the magnetotail, where field lines cross creating neutral lines. The solar plasma is then channelled back into the inner magnetosphere.

When the solar wind magnetic field is in the opposite orientation as the Earth's, a magnetic reconnection occurs (see Figure 2.2). The magnetic fields of the Sun and the Earth are combined together, which opens the magnetopause. As a result, the solar charged particles enter more widely into the atmosphere, mainly around the poles. Once a particle is inside the magnetosphere, it is trapped and bounces back and forth between the two poles. All these trapped particles form the Van Allen radiation belts. Charged particles can also be stored in the magnetotail, in a region called the plasma sheet. If a particle has sufficient energy, it comes into the upper atmosphere, thus creating auroral emission.

When a magnetic reconnection occurs, it causes a geomagnetic storm, which is a fluctuation of the Earth's magnetic field. During this event which can last for around a day, the magnetotail is distended, thus strengthening the electric fields in the magnetosphere and

ionosphere. This can lead charged particles to enter the thermosphere and create strong auroras.

2.3 Auroral phenomena

The brightest auroras are seen during what is called auroral substorms. They require an unstable magnetotail along with strong electric currents in the magnetosphere. This can be provided by geomagnetic storms, but also with smaller disturbances associated with a South-North polarized solar wind. Indeed, high-speed solar wind often switches between northward and southward polarisation when arriving towards Earth and can produce many short substorm events. High speed solar wind occurs mostly during the minimum of solar activity, whereas CMEs and geomagnetic storms are more likely during solar maxima.

During auroral substorms, a portion of the electric current across the magnetotail is shunted through the auroral ionosphere. Electrons stored in the magnetosphere are propelled downwards into the thermosphere ionosphere, and upper mesosphere (from 350 to 80 km above Earth's surface). When the particles enter the atmosphere, they collide with atoms of Oxygen and Nitrogen. The energized atoms then emit photons with a pure color when they return to a lower energy state. Photons emitted from Oxygen atoms are mostly in the green and red wavelengths. Nitrogen atoms produce blue light.

Auroras occur mostly in the two auroral ovals, centered around the magnetic poles. These ovals vary in size along with solar activity. They are up to four times wider on the night side. Auroras on the day side are much weaker, as they are only caused by particles which were already trapped in the magnetosphere. Another type of weaker aurora called the polar cap auroras can occur when electrons get in through the far end of the magnetotail. It happens inside the auroral oval.

2.4 Evolution of auroral substorms

Even though auroral displays always differ from one another, some common characteristics can be found [3].

An auroral substorm has a lifetime of around 1 to 3 hours. It can be divided into 2 phases, the expansive phase lasting 10 to 30 minutes and the recovery phase which can last more than 2 hours.

2.4.1 Quiet phase

Between substorms, during what is called the quiet phase, auroras are seen in the form of dim arcs oriented in the East-West direction. Polar cap auroras can also be seen in quiet conditions; they are perpendicular to the auroral arcs.

2.4.2 Expansive phase

In the first 5 minutes of the expansive phase, one of the quiet arcs around the midnight meridian becomes suddenly brighter. In the next five minutes, the arcs observe a poleward motion, creating a great bulge centered around the midnight meridian. This creates folds in the auroral arcs and can quickly fill the sky with bright bands, into what is called a breakup aurora. The last stage, which lasts from 10 to 30 minutes, is characterized by eastward and westward motion (respectively in the morning and the evening sides) of

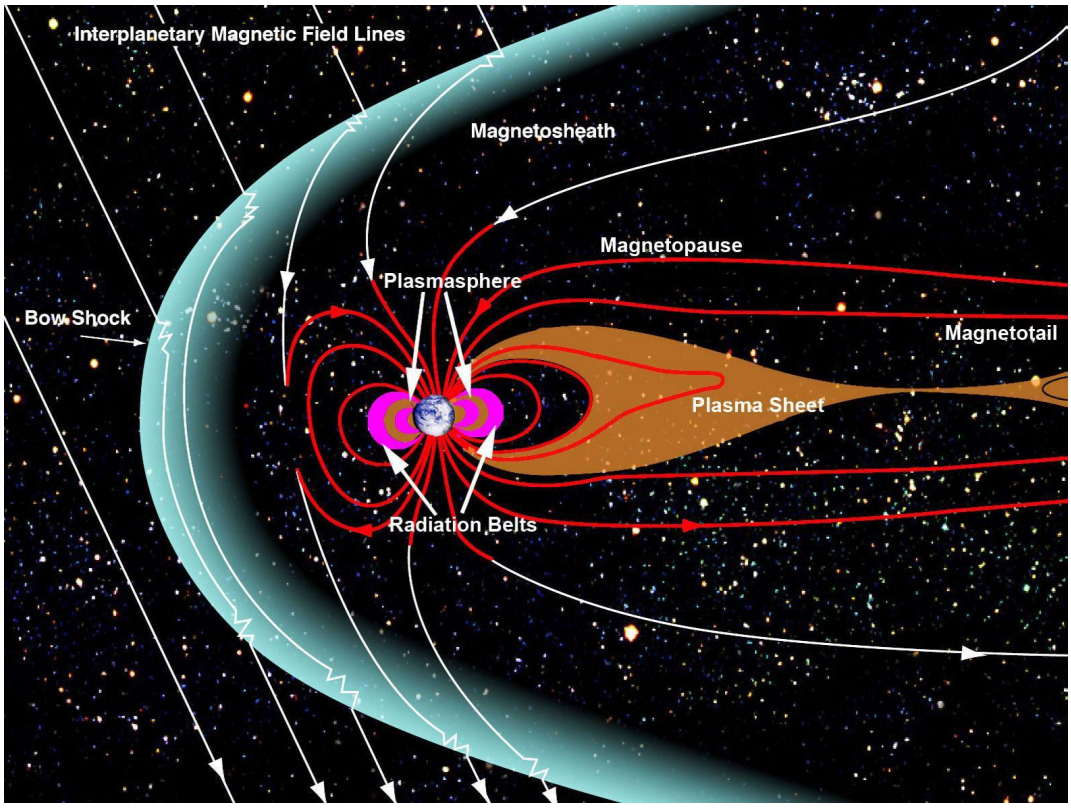


FIGURE 2.2: Illustration of the magnetic reconnection of the Sun's and the Earth's magnetic fields. (NASA / Goddard / Aaron Kaase)

active or broken up bands. In the morning sector, distinct bands appear which are called omega bands. Equatorward, broken up arcs appear as patches of isolated cloud-shaped auroras, which will in term drift eastward and westward.

2.4.3 Recovery phase

During the first hour of the recovery phase, the bulge begins to reduce in size, towards its initial position. In the evening sector, well-defined loops appear, drifting westward. In the morning sector, most arcs and bands disappear and auroral patches are widely spread. In the next hour, new arcs tend to reappear in the polar cap and in the patches. The drifting loops become more irregularly shaped. The last stage consists in a slow equatorward motion of arcs. No more drift is observed in the evening and morning sectors. This stage constitutes the transition between either the next auroral substorm, or a quiet phase.

3 Description of unsupervised classification processing chain

3.1 Data sources

In the literature, data sources for auroral image can be very diverse on a number of levels. Firstly, all acquisition instruments are located in ground-based observatories. These are spread in the northern polar region, in different countries at high latitudes, in Alaska, Canada, Norway and Sweden. Some of the northernmost observatories are located on the Spitsbergen island at Ny-Ålesund. Depending on the latitude of observation, the viewing angle on the auroral oval will be different. Thus, there are differences in shape and position of auroras between locations on separate latitudes. In addition, observatories located at high latitudes are able to observe auroras during the day in the winter season. Indeed, because of the tilt of the rotation axis of the Earth, the Sun is always set during the winter season. Thus, high-latitude locations can be exploited to observe dayside auroras in particular, as it is impossible to do so at lower latitudes.

Differences in data sources also come from the acquisition instruments. The wide majority of imaging systems consist of optical photographic devices equipped with fish-eye lenses offering a full-sky view. Most devices provide grayscale images. These images come from filtered acquisitions at specific wavelengths corresponding to the monochromatic emissions of auroras. For instance, the Arctic Yellow River station at Ny-Ålesund uses a filter for the 557.7 nm wavelength emission, corresponding to the dominant green color of auroras [4]. This allows to avoid perturbations from other light sources emitting in different wavelengths. However a part of the information is lost, because auroras also emit in blue and red wavelengths. Some observation centers, such as the Kiruna observatory in Sweden[5], or the Kjell Henriksen observatory[1] include color cameras which can provide information from the whole optical spectrum. Such images are however often polluted by other light sources such as ground light pollution, clouds, the moon, or the sunlight scattering in the atmosphere.

The predominant sources of perturbation on the images are the moon and the clouds. Therefore, in most automatic classification contexts, images containing either the Moon or clouds are excluded beforehand. This allows to work with clean images which will be easier to use for machine learning. The sorting process can be made manually using keograms[1] or using automatic selection based on additional information, on the Moon's position for instance, or using an automatic classifier to detect the presence of auroras.

All-sky images are captured repeatedly over long time periods. Rates of acquisition vary from 30 seconds to a few minutes. This way, more information can be used for studying the auroral evolution, as auroras can evolve over short timescales. However a high rate of acquisition can produce series of very similar images during quiet periods of magnetic

[1]Keograms consist in taking the central column of each image of a time sequence and building a new image indicating the presence of auroras or clouds over time.

activity. In such cases, a careful data selection must be made. Indeed, machine learning algorithms must have a sufficiently varied training dataset along with a distinct testing dataset which must be representative of the whole dataset. To achieve this goal, one can either sub-sample the data or split specific parts of the data using cross-validation for instance.

3.2 Choice of auroral classes

Most of the data sets created for auroral morphology classification have labeled images, indicating in each image the shape, texture or brightness of the aurora, or the presence of other objects such as clouds or the Moon. These labels allow to have a ground truth reference which will serve as a basis of machine learning algorithms for automatic classification of auroral morphology. We will see that the categories which have been established are very diverse, and that the research community does not always agree on the right categories to define.

One of the first works which establishes an aurora classification based on photographs is the International Auroral Atlas published in 1963 [6]. It relies on images from various locations, different viewing angles and can be in color or grayscale. The classification criteria were the form, structure, condition, brightness and color of auroras. Each criterion allows to create a number of classes within it. Auroral forms can be *Arcs* (thin curved bands), *Bands* (Bands with more irregular shapes), *Patches* (isolated cloud shapes), *Veil* (Uniform intensity across a large part of the sky), *Rays* (luminosity rays along the geomagnetic field), and other non-identifiable forms. Structure characterizes the texture of the aurora, which can be *Homogeneous*, *Striated*, or *Rayed*. The condition criterion refers to the temporal evolution of the aurora, which can be *Quiet* if it remains stationary, *Active* if it changes quickly over time, or *Pulsing* if the auroral brightness fluctuates repeatedly. Brightness and color are respectively classified into 5 and 6 classes. As a result, many classes are formed by combining each criterion. However, the data used for this atlas is not uniform in terms of observation conditions and sometimes relies upon temporal information, thus a more rigorous approach needs to be found.

One way to categorize auroral forms can be to follow the description of the evolution of auroral substorms [3]. During quiet phases auroras are often seen as arcs crossing the sky in the East-West direction and are labeled *Arc auroras*. During the expansive phase, the arcs are folded and begin to take the whole sky, thus creating *Breakup auroras*. Then, in the morning sector appears the *Omega bands aurora* named after their distinctive shape. In the same time frame, cloud-shaped auroras appear as isolated dim patches, thus named *Patchy auroras*. During the recovery phase, arcs take more irregular shapes, which can therefore belong to the *Irregular auroras*.

One must however keep in mind that this classification only applies to nightside auroras and can be further improved by having a more detailed description of the phenomena from which the auroras originate.

In [7][8], researchers used The Oslo Auroral THEMIS dataset (OATH)[8] which use panchromatic imagery (producing non-filtered grayscale images). In this dataset, images are labelled into 6 categories (Figure 3.1): *Arc*, *Discrete*, *Diffuse*, *Cloudy*, *Moon* and *Clear/No aurora*. The *Arc* class corresponds to sharp bands which extends across the sky. The *Diffuse* class consists of large fuzzy structures. The *Discrete* class regroups sharp auroral shapes but which are not arc-like. The *Cloudy* class indicates the presence of clouds or snow on

the observation dome. When the light of the Moon dominates the image, it is sorted in the *Moon* class. The *Clear/No aurora* class corresponds to images with clear sky where we can see the stars without aurora.

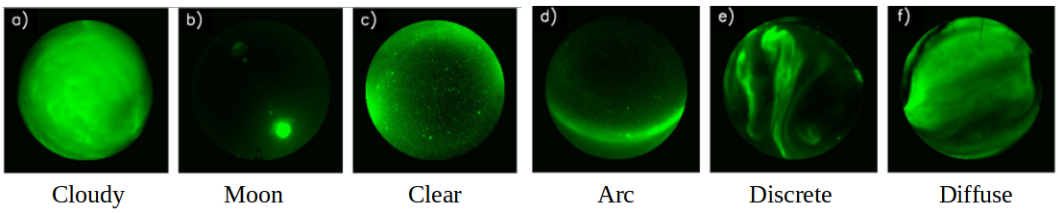


FIGURE 3.1: Samples from each class of the OATH dataset. The green color is false, as these are grayscale images. [8]

In one study [5], 14000 color images from 9 years of observation at Kiruna in Sweden were labelled into 7 classes of aurora (Figure 3.2): *Auroral breakup* (large and bright auroras), *colored auroras* (when there is a prominent red, blue and purple emission), *Auroral arcs* (East-West arc-shaped auroras), *Discrete-irregular* (Distinct auroral shapes which are not arc-like), *Patchy auroras* (dim and diffuse irregular shapes), *Edge aurora* (when auroral emission occurs at the edge of the image) and *Faint/Clear* (dark images with no visible auroras).

FIGURE 3.2: Samples from each class of the dataset used in [5] (color images)

Most studies dealing with auroral morphology classification already rely on pre-established classes of auroral forms. One way to classify auroral morphologies is to automatically learn the distinctive features of auroral forms in order to find the meaningful groups in a wide set of unlabeled images, and thus define new categories of auroral morphologies. This allows to avoid human bias which can occur while using pre-established classes. In addition, we know that the pre-established classes cannot encompass the whole diversity of auroral forms, where for instance in one study [9] only 12% of auroral images could be assigned to a defined class. Automatically finding new classes could allow to better take into account this diversity.

3.3 Overview of the main processing steps

To achieve the goal of finding pertinent auroral classes using a dataset with no predefined labels, four main processing steps can be used. The methods listed here have not always been applied to the exact same issue. However, they are in majority applied to the same kind of auroral images as those available for this project. The state-of-the-art mainly consists in supervised auroral classification and search engine for similar auroral images, whereas only two studies have tackled the issue of unsupervised clustering of auroral images [4][10].

The first step is the preprocessing of the images. It allows to eliminate unwanted information, such as text on the image, or the stars, and to reduce biases in the data induced by the background illumination and hue for example. It can also allow to improve the

image quality by enhancing the contrast for instance. These preprocessing methods need to be adapted to the processing steps of the data in this project.

The second processing step consists in extracting the main features from the images of the data set. This step gives a feature vector as an output for each image. This can be done in various ways, which can be separated into 2 groups: manual feature extraction and automatic unsupervised feature extraction. In the first category, one can use global statistical image descriptors such as mean, standard deviation, or maximum of pixels intensities or color. To capture more abstract information such as shape or texture, more advanced local descriptors can be used such as the Gray-level Cooccurrence Matrices (GLCM) or Scale Invariant Feature Transform (SIFT). In this case, we need to manually select which features are the most pertinent, which is a difficult challenge.

Another approach for feature extraction is to learn automatically the most pertinent features based on the data set, by using deep learning methods. In this category, because we do not have predefined labels, two main options can be used. The first one is to use Pretrained Neural Networks as feature extractors. These networks are trained on standard data sets based on different classification tasks [11], but the features they can extract are general enough to be reused in other contexts. The other option is to use unsupervised feature learning methods. Their general principle is to define an annex goal for which we can automatically provide labels, in order to train a feature extractor which can be readily used for the initial task. One example is the Autoencoder Network, which aims to build a compressed representation of an image and reconstruct the original based on this representation. The encoding part can then be used as a feature extractor. Siamese networks such as SimCLR can also allow to extract image features without labels. We first need to create a set of distorted images for each original image of the data set. Then the network is designed to find from which initial image the distorted one comes from.

The third processing step is the reduction of dimension of the feature vectors. This avoids having high dimensional vectors at the clustering step, which can be problematic [12]. This step is not always necessary, as it depends on the output dimension of the feature extraction step. For instance, Autoencoders already provide a low-dimensional feature vector. The most common operator is the Principal Component Analysis in its linear form, or non-linear form by using a kernel function. Other non-linear methods include Isomap, UMAP, t-SNE, or Multi Dimensional Scaling, which rely on topology notions to keep the initial structure of the data while reducing the dimension.

Once the dimension of the feature space is sufficiently low, the final step consists in finding clusters in the data set. To achieve this goal, there are a wide range of algorithms that can be used. The most common one is the K-means algorithm, but depending on the shape of the clusters we want to find, other algorithms such as Spectral clustering can be more pertinent. To achieve better performance, additional information on some the clusters can be provided, which is called constrained clustering. Simple clustering methods can also be combined, by using voting mechanisms for instance. This family of clustering methods are called ensemble clustering.

3.4 Preprocessing

The goal of the preprocessing step is to avoid any perturbation which could impact the classification. This step allows to drop the excess of information present in the initial image, to decrease the amount of noise in the data, and to normalize the data in regard

to the whole dataset. In the case of auroral images, some specific preprocessing steps are used.

In one study [5], a median filter with a 3x3 pixel kernel is applied in each image to delete the stars, which are generally the size of one pixel when imaged. This allows to reduce acquisition noise from the camera, for instance thermal noise from the sensor. To remove bright objects such as stars, one can also scale the intensities of the image based on the intensity percentiles, which in addition allows to obtain a smoother background. In [7], the preprocessing step consists in linearly scaling the images intensity between the 0.5 and the 99.5 percentiles. In [10], the image intensities are adjusted and normalized according to the following function:

$$I_n = max(min(\frac{I - Q_1}{Q_{99}}, 1), 0)$$

where Q_1 is the first percentile of intensities of the image I, and Q_{99} is the 99^{th} percentile of $I - Q_1$. In this case, the resulting image intensities are normalized between 0 and 1, which is an important condition for further use in a machine learning context. However, depending on the network used, especially when using pretrained networks, the normalization step can be adapted, for instance by having image intensities between -1 and 1.

Another important step in preprocessing auroral images, is to take into account the specific round shape of all-sky images, which come from the fish-eye lenses being used. Two studies [10][5] have implemented a step consisting in cropping the center of the image to exclude the black corners in the image. This also allows to exclude images containing auroras which are near the horizon and thus difficult to identify as their shape is highly distorted.

Other image processing algorithms can be used to enhance the images and make them more uniform across the dataset. To do so, one can use color histograms transformations. Directly transforming the Red, Green and Blue channels of a color image would transform the image too much and meaningful information would be lost. This is why transformations are usually made in more meaningful color spaces, such as the Hue, Saturation, Value color space. For instance, this space allows to only work on the overall brightness of the image with the Value channel and use the aforementioned percentile adjustments. We can also perform an histogram stretch on the Value channel to enhance the contrast of the image. The other channels can also be used to adjust the colors, to avoid classification biases when the background sky can have different hues for instance.

3.5 Features extraction

3.5.1 Manual features

In the context of automatic processing of auroral images, numerous methods of feature extraction have been used in the literature. The goal is to extract meaningful information from an image in order to form a vector which will synthesize this image. This allows to avoid having redundant information and thus a too high amount of data. However, feature extraction is a difficult task because of the wide range of image features to choose from. Feature extraction methods for auroral classification can be divided into three main groups, which are contour-based, texture-based, or multi-resolution methods.

Contour-based methods consist in extracting the contours or edges of the aurora and then to use the shape descriptors to characterize the auroral shapes which are present in the

image. For instance, in a paper from 2007 [13], Fourier shape descriptors are used which are able to accurately describe previously extracted contours. These descriptors are used to form feature vectors which can be compared using the euclidean distance.

Another type of image characteristics is the texture information. Indeed, some auroras can be differentiated not only by their shape, but also by their texture. This can tell if they are homogeneous or more irregular for example. The first texture-based method applied to grey-level auroral images is based on Basic Grey-Level Aura Matrices (B-GLAM)[13]. The principle is that the GLAM of an image characterizes the probability distribution of each grey level in the neighborhood of each other grey level. It is a generalisation of the Grey-Level Co-occurence Matrices (GLCM). B-GLAM is a variation of GLAM which has the interesting following property: the B-GLAMs of 2 images are the same if and only if the images are the same [14]. A distance function based on B-GLAMs is then used to compare images based on their texture in a supervised classification context.

Another more recent approach [15] is to use the Local Binary Patterns as a descriptor of texture. In this method, the neighborhood of each pixel is evaluated, by comparing each grey-level value of neighbor pixels to the pixel in the centre. Each configuration of neighborhoods is coded into a binary number. Then, the information is summarized using the histograms of the binary patterns, which are then used as feature vectors.

A third category of feature extraction is based on multi-resolution analysis. The most widely used method in this category is the Scale Invariant Feature Transform (SIFT) method [16], which has been used in the auroral image feature extraction context [15]. The multi-scale aspect of this method lies in the first step, which is the scale-space extrema detection. It computes the Difference of Gaussians (DoG) of the image, which is obtained through subtracting consecutive Gaussian-filtered images at different scales. Local extrema are then found in the scale space of the Gaussian pyramid. After a selection step to only keep interesting key-points among the computed extrema, for each key-point a gradient orientation is computed. Orientation of the neighbor pixels are summarized into an orientation histogram for each key-point. These key-point descriptor can then be used as feature vectors for the auroral images.

3.5.2 Deep learning methods

Because finding meaningful features is a very challenging tasks, a large part of modern literature focuses on methods which are able to automatically extract meaningful features using deep learning. This is especially justified since large amounts of data from all-sky imagers are available, and since the features involved in auroral classification are varied and complex.

In the context of image features extraction, a widely used and effective deep learning algorithm is the Convolutional Neural Network (CNN) [17]. It is especially adapted to image processing, in supervised or unsupervised contexts. Its principle is based on standard feed-forward neural networks like the Multi-Layer Perceptron (MLP). However instead of computing a weighted sum over the preceding neurons and working with vectors at each layer, CNNs use sets of small trainable kernels which are applied to the image using the convolution operator. An activation function is usually applied to the results of the convolution operations in order to produce a non-linear output. The sets of convoluted images are then passed along each convolutional layer in a feed-forward style. Depending on the objective, the number and size of kernels at each layer can vary. In addition, in order to capture features at different scales, the size of the filtered images can be reduced between some layers of the network. Many variations exist on the basis

of convolutional layers. As the dataset for the current project is not exhaustively labeled, we will only focus on methods which can be applied in an unsupervised context.

Pretrained neural networks

Pretrained networks are artificial neural networks which have already been trained on a dataset which is different from the dataset of interest. Typically, pretrained networks are trained on a standardized dataset of diverse images called ImageNet [11]. These networks are trained to classify images into 1000 classes representing common objects or animals such as 'husky', 'coffee mug', or 'container ship'... Even though the dataset of interest is different, it is still pertinent to use such networks in order to extract the image features. A parallel can be made with human perception: when trying to describe clouds for instance, we use comparisons with known objects (an elephant, a face, etc.) which do not directly relate to clouds in the first place.

The use of pretrained networks is justified by the expensive training of conventional CNNs. In supervised classification contexts, by only using a pretrained network as a feature extractor, it is possible to train a simple classifier at the end such as a Support Vector Machine or a Ridge Classifier [7]. In the case where the amount of auroral data available is small, it can avoid overfitting, because the data is not used to update the parameters of the complex pretrained network. Another advantage is that feature extraction can be done once; the feature vectors are stored once and can be reused for training simple classifiers. This operation can thus save computing time. These networks can also be used in unsupervised contexts, as they do not require any additional training for feature extraction.

Many pretrained networks exist in the literature, with improved performance over the years. We will focus here on only 3 architectures, which rely on 3 distinct concepts: Resnet, Inception and Mobilenet. Even though they are not the highest performing networks, they are still reliable and easily usable as they have relatively low computational costs. They are also mentioned in auroral classification contexts [10][7][5].

The Resnet architecture [18] has been designed to solve the "vanishing gradient" issue [19] which happens when using deeper network architectures. During the back-propagation of the error through the weights of each layer, the gradient of the loss function can become extremely small, with an exponential decreasing rate on earlier layers.

The novelty of Resnet architectures compared to traditional convolutional networks is the introduction of skip connections (see Figure 3.3). The principle, instead of simply stacking convolution layers, is to sometimes add the original input to the output of a convolution layer, or a group of layers. This allows the gradient to better back-propagate through the layers. Thus, deeper architectures can be used without performance loss.

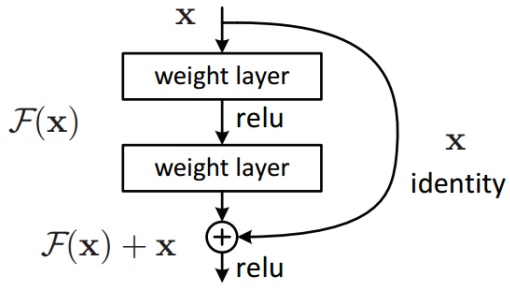
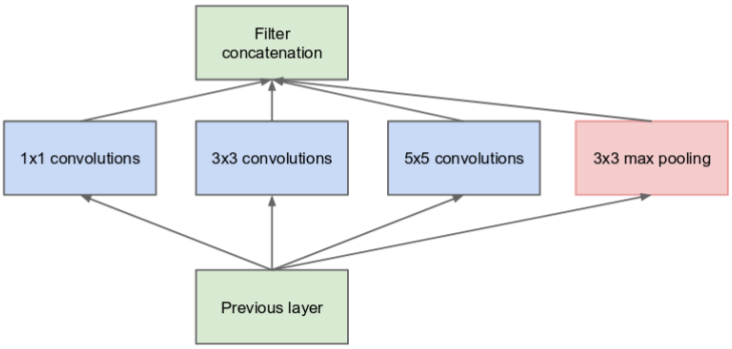


FIGURE 3.3: Theoretical building block of the ResNet architecture [18]

In Resnet, one building block is composed of a convolution layer followed by an activation function (in this case the ReLu function[2]), and another convolution layer. Before passing through another activation function, the input of the block is added to the output. These building blocks can be used to build various architectures, having different depths which are shown in their names: Resnet-50, Resnet-34, etc.

The Inception network architectures [20] are based on one principle. In order to capture distinctive image features at different scales, multiple convolution operations with different kernel sizes are applied to the same input (Figure 3.4). In addition, a max pooling operation is applied. All the outputs are then concatenated to form the block output. Such building blocks are then combined to create full architectures. The first version of Inception has been improved several times in order to improve the computational complexity and performance.



(a) Inception module, naïve version

FIGURE 3.4: Building block of the basic Inception architecture [20]

The Mobilenet architectures aim at being more computationally efficient, in order to be usable on portable devices such as smartphones [21]. The main concept here is to replace traditional convolution operation by depth-wise separable convolutions (Figure 3.5), which are less computationally expensive. In a normal convolution operation, if the input has a size of $(A \times A \times M)$ with M the number of channels (for instance 3 if it is the input color image) and if there are N kernels of size $(B \times B \times M)$, then the output size will be of size $(C \times C \times N)$ with $C = \lfloor \frac{A}{stride} \rfloor - \lfloor \frac{B}{2} \rfloor$. The stride corresponds to the step with which the kernel moves along the input.

In depth-wise separable convolutions, the operation is split in two. The first step is the depth-wise convolution. This time, one filter is applied to each input channel. Therefore, there are M filters of size $(B \times B \times 1)$. The result is then of size $(C \times C \times M)$. The second step is point-wise convolution, in which a 1×1 convolution is applied on the M channels. In this case, we have N kernels of size $(1 \times 1 \times M)$. Applied to the output of the first step, we obtain a final output of size $(C \times C \times N)$. The amount of operations is reduced by a factor of $\frac{1}{N} + \frac{1}{B^2}$. Thus the computational advantage of depth-wise separable convolutions increases with the number and size of the filters.

Unsupervised learning

Autoencoders are deep-learning architectures which consists in approximating the identity operation while producing a compressed version of the initial object in the hidden

[2]The ReLU function is expressed as: $ReLU(x) = max(0, x)$

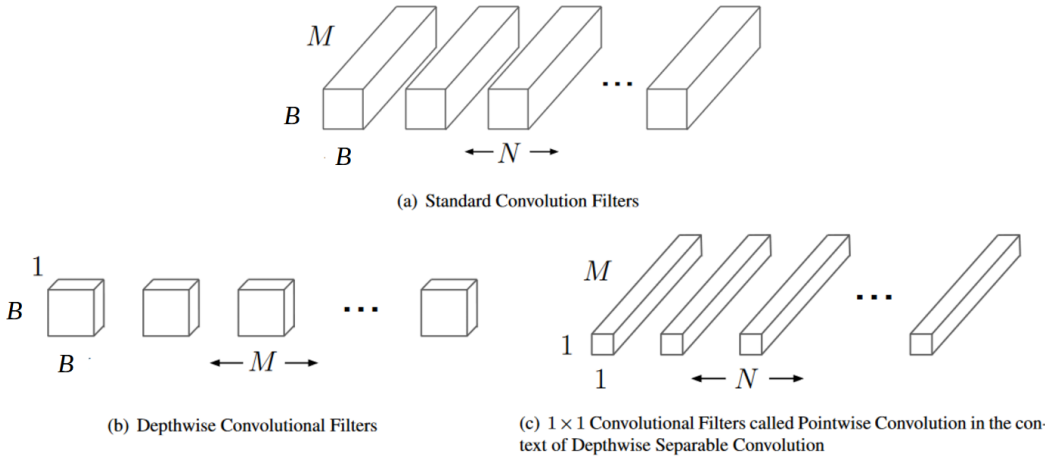


FIGURE 3.5: Illustration of the depthwise convolution principle [21]

layers of the network. It has been used in 2021 [4] for unsupervised classification of auroral images.

Most autoencoders consist of two parts: the first is the encoder which is a neural network with its layers progressively decreasing in size. Its output is a compressed version of the input object. The second part is a decoder, which takes the compressed vector as input, and with increasingly big layers, its output lies in the same space as the input object. The training objective is simply to obtain an output which is the most similar to the input.

The autoencoder is then used as a feature extractor or a dimension reducer, by using only the encoding part. This architecture is trained in a self-supervised manner, which means because the label is the input object itself, no extra label is needed.

For image features extraction, a convolutional version exists, called the Deep Convolutional Autoencoder. The neural layers of decreasing dimension are replaced by convolution layers with max-pooling[3] layers in-between, and neural layers increasing in size are replaced by transpose convolution layers[4]. These correspond to a convolution which increases the output size of the image.

The second method used for unsupervised classification of auroral images [10] is a Siamese network architecture called Simple framework for contrastive learning of Representations (SimCLR)[22]. The goal of this unsupervised network is to be able to tell if two randomly distorted images come from the same sample. In the process, it learns meaningful features which can effectively represent each sample and be comparable to features from other samples. This network is composed of 4 successive steps:

The *data augmentation module*, which applies 2 different random transformations to a sample image. This will give a positive pair, as they are derived from the same sample.

The *base encoder*, a network which extracts a representation of each transformed input. In general, networks such as ResNet are used, which can be pretrained or not.

[3] A max-pooling operation consists in taking the maximum pixel value in a small sliding window, usually of size 2x2. This allows to reduce the dimension of the image by a factor of 2 in height and width.

[4] The principle of a transpose convolution layer is to multiply each input value by a filter, which gives an output with the size of the filter. Each output is usually assembled with a stride of 2, thus creating an output image with a doubled size.

The *projection head* which maps the output to the space where contrastive loss will be applied. It is a machine learning model which is most often a shallow neural network, consisting of one or two dense layers.

The *contrastive loss function*, which is designed to follow the goal of identifying the positive pairs among other transformed samples in a batch. In [10], the function used is the *normalized temperature-scaled cross-entropy loss*:

$$l(x_i, x_j) = -\log \frac{\exp\left(\frac{sim(z_i,z_j)}{\tau}\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\frac{sim(z_k,z_i)}{\tau}\right)}$$

$$where\ sim(x, y) = \frac{x^T y}{||x||.||y||}$$

with z_i, z_j the outputs of the projection head for each transformation, $\mathbf{1}_{[k\neq i]}$ equals to 1 if $i \neq k$ and 0 otherwise, N is the minibatch size, τ is the temperature parameter.

Because this loss is asymmetric, the model must minimize the following symmetrised function:

$$\frac{1}{2N} \sum_{k=1}^{N} [l(2k-1, 2k) + l(2k, 2k-1)]$$

In [10], the chosen transformations are a random resized cropping and a random horizontal flip. Vertical flip is not considered as there is a loss of location information, because auroral structures are organized along the East-West axis, which is the horizontal axis. Random color transformation is abandoned, as it reduces the pertinence of the clusters found in the following steps, even though it improves performance of the feature extractor.

3.6 Dimension reduction

Before using clustering methods to find groups in the feature space, a usual step to include is the reduction of dimension. Indeed, in some cases the feature vectors are expressed in a high-dimensional space, and clustering algorithms based on the Euclidean distance for instance can be less effective [12]. One solution to this issue is to reduce the dimension of the feature space. This leads automatically to a loss of information as less variables are captures. But the data does not always have high variations along all dimensions, and in most cases a lower dimensional space can be found in which only a small amount of information is lost. We will focus here on four main dimension reduction techniques which are among the most popular in the scientific community : Principal Component Analysis (PCA), Isometric mapping (Isomap),t-Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP).

3.6.1 Principal Component Analysis

The Principal Component Analysis (PCA) is a linear dimension reduction method which consists in finding orthogonal axes in the multidimensional space which maximize the variance of the vector data. These axes form an orthogonal basis on which the vector data can be projected. The dimension is reduced when the data is projected only along the axes representing the most variance. The main mathematical computation behind PCA is to find the eigenvectors of the covariance matrix of the centered dataset matrix. Dimension

reduction occurs when only a direction of the eigenvectors are taken, corresponding to the highest eigenvalues [23].

In order to take into account non-linearity in the data organisation, there exists a non-linear version called Kernel-PCA. Its principle is to virtually project the dataset into a high-dimensional space using a kernel function. This method is useful in the case where groups of data are not linearly separable e.g. nested spheres, because it is easier to find separating hyperplanes in high-dimensional space[24].

3.6.2 Topological methods

More complex dimension reduction methods exist, which rely on concepts of mathematical topology and better take into account non-linear organisation of groups of data.

Isometric mapping

One such method is Isometric mapping (Isomap), developed in 2000 [25]. It relies on the geodesic distance, which is the shortest path along a given manifold. This allows to better represent data organized in folded patterns . The main example is the "swiss roll" data set (Figure 3.6). The Isomap algorithm is the following:

- First, build a graph from the data points by connecting two points if they are close together. This criterion can be either to connect 2 points if they are closer than a distance ϵ, or if one of them is in the k-nearest neighborhood of the other. The length of each edge of the graph is the Euclidean distance between the corresponding two points.
- Then, compute the shortest path between each point along the graph. Multiple algorithms can be used for this computation, such as the Floyd-Warshall algorithm [26]. Each value is an approximation of the geodesic distance along the constructed graph. The results are put in a distance matrix D.
- The last step is to build a low-dimensional embedding from the distance matrix:
  - First, take the matrix S of squared distances: $S_{ij} = D_{ij}^2$
  - Then apply double centering: $B = -\frac{1}{2}CSC$ where C is the centering matrix: $C_{ij} = \delta_{ij} - \frac{1}{N}$, and N is the total number of images.
  - Determine the m highest eigenvalues and the corresponding m eigenvectors of B, with the desired number of dimensions of the output.
  - Then the final data matrix is: $Y = V_m \Lambda_m^{1/2}$, with V_m the matrix composed of the m eigenvectors and Λ_m the diagonal matrix containing the m eigenvalues.

t-distributed Stochastic Neighborhood Embedding

A popular tool for visualising high-dimensional data is t-distributed Stochastic Neighborhood Embedding (t-SNE) [27]. The first step of the algorithm is to compute similarities between each pair of points x_i and x_j as follows:

$$if\ i \neq j : p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

$$if\ i = j : p_{j|i} = 0$$

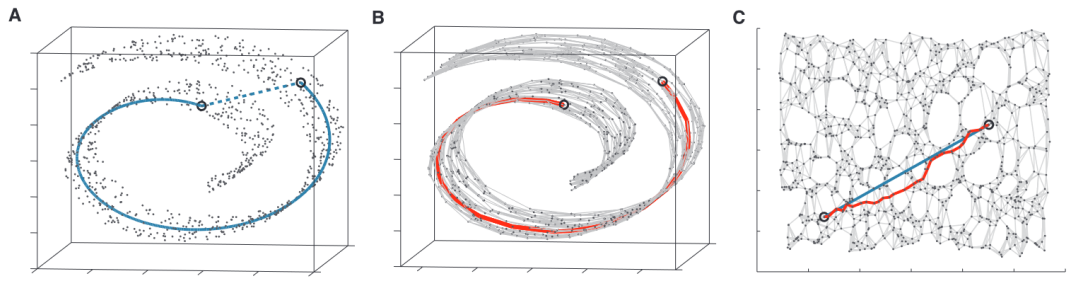


FIGURE 3.6: Illustration of the Isomap principle on the "swiss roll" 3-dimensional data set. On the left we see the euclidean and geodesic distance between to points. In the center we see the approximation of the geodesic distance computed with Isomap. On the right we see the 2-dimensional mapping of Isomap which preserves the geodesic distance as the shortest path between the 2 points. [25]

This similarity function $p_{i|j}$ corresponds to the probability to pick x_j as the neighbor of x_i if it was following a Gaussian probability density centered on x_i. The factor σ_i is the bandwidth of the Gaussian functions and is computed so that the perplexity of the conditional probability given the data point x_i noted P_i, matches a user-defined constant. Perplexity is defined as:

$$pp(P_i) = 2^{-\sum_j p_{j|i} log_2 p_{j|i}}$$

This can be seen as a measure of the number of effective neighbors.

In order to have a symmetrical similarity, we set:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

With N the total number of data points.

The goal of t-SNE is to build a map of N vectors y_i with m dimensions which best matches the similarities p_{ij}. To do so, we define a similarity function in the target space as follows:

$$if\ i \neq j : q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||y_k - y_l||^2)^{-1}}$$

$$if\ i = j : q_{ij} = 0$$

In order to match the two probability distributions P and Q, we need to minimize the Kullback-Leibler divergence:

$$KL(P|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The minimisation is then done using a gradient descent algorithm.

Uniform manifold approximation and projection

A more recent variation of t-SNE is the Uniform manifold approximation and projection (UMAP)[28]. It uses the same principles as t-SNE: a k-neighbors graph is constructed from the data, then an objective function is optimized to find a low-dimensional map which is most similar in structure as the high-dimensional data.

The main difference with t-SNE is that the similarity measure in the high-dimensional space is locally adapted to the density of each neighborhood. In addition, no normalisation is carried out in both high- and low-dimensional similarity measures. Instead of the Kullback-Leibler distance, the cross-entropy is used:

$$CE = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1 - p_{ij}) \log \frac{1 - p_{ij}}{1 - q_{ij}}$$

With these changes, the UMAP algorithm is not only more computationally efficient than t-SNE but also more capable of preserving global structure information [28].

3.6.3 Issues with t-SNE and UMAP

One must however keep in mind that in the case of auroral image classification, t-SNE and UMAP are used for visualisation only [7][5]. It has been shown for instance that t-SNE can be used for clustering applications [29], but in general, their extension to such applications come with a number of issues.

In the case of t-SNE, because the global structure of the data is not preserved, the distances between dense areas and their size do not reflect the initial global density distribution. Therefore distance and density-based clustering methods applied to t-SNE results could be biased [30].

Both t-SNE and UMAP rely on hyperparameters (respectively the perplexity and the number of neighbors) in the computation of high-dimensional similarity which can highly impact the aspect of the result [30]. The stochastic nature of both algorithms imply that different results can occur from identical conditions, thus multiple runs with the same hyperparameters can be useful. The last issue is that these algorithms tend to create artificially dense regions even when only noise is present in the original data set, depending on the choice of the hyperparameters [30].

3.7 Clustering

The objective of the clustering step is to find a partition of a data set in which we have distinct groups. These groups ideally reflect areas of higher density and are not overlapping. A wide range of simple clustering methods exist, each one having its strengths and weaknesses. Therefore, more advanced methods such as Ensemble clustering combines different clustering methods to compensate the weakness of each individual method, thus allowing more robust solutions. Another possible solution is to use specific constraints in the clustering process based on context knowledge, which will help to find more pertinent solutions. However the main issue of unsupervised clustering methods is the lack of reliable performance metrics. Indeed, because we do not have access to a ground truth in this context, typical accuracy measures cannot be applied. However other methods exist, such as internal validity scores or stability analysis which can provide some insight into the quality of a clustering result.

3.7.1 Simple clustering

K-means clustering

The principle of the K-means algorithm (also called Lloyd's algorithm) is to compute K centroids so that each sample point (corresponding to a feature vector) is assigned to

its nearest centroid, thus defining K clusters [31]. The goal is to find values for the K centroids which will minimize the distance between point belonging to the same cluster. More precisely, it searches to minimize the sum of distances squared between each sample point and its respective cluster centroid. This value is called the *inertia*. The steps of the K-means algorithm are the following (we assume that the number of cluster K is defined):

- First, randomly choose the position of the K centroids among the sample points of the dataset.
- Assign each sample point to its nearest centroid, by means of the Euclidean distance.
- Shift each centroid to the mean position of its cluster.
- Repeat the last two steps until the centroids do not move significantly anymore.

To avoid falling into local minima, centroids are usually set so they are sufficiently far from each other. This adjustment is called *K-means++*.

The major assumption of the K-means algorithm is that the computed clusters are convex and isotropic: This means in case the data is organized into complex folded or elongated clusters, the clustering can be less performing. The inertia criterion of K-means uses the Euclidean distance, which can be an issue when the data is high-dimensional. Indeed, when the number of dimension is high, the Euclidean distance tends to become less discriminative. This means that distant points tend to have a distance value similar to the distance between points close to each other, when the number of dimensions increases. To avoid this issue, also called the "curse of dimensionality", reduction of dimension can be used, if low dimensional structure can be found in the data [12].

Spectral clustering

Another method which we can use is called spectral clustering [32]. Spectral clustering is a popular method which is simple and adaptable. It is however limited in scalability and generalisation.

The principle of spectral clustering is to solve the spectral decomposition of the similarity matrix to make the intra-cluster similarity and inter-cluster difference of the objects large. The goal is to first represent objects by their eigenvectors obtained by spectral decomposition, then cluster the eigenvectors using a regular k-means algorithm.

To perform spectral clustering, we need first to build the adjacency matrix A, from the graph corresponding to the cloud of vectors we need to cluster. This matrix is built using the Euclidean distance, thus $A_{ij} = A_{ji}$ corresponds to the distance between vector i and j. Then, we compute the Laplacian matrix of the graph:

$$L = D - A$$

where $D_i = \sum_j A_{ij} = \sum_j A_{ji}$ is a diagonal matrix called the degree matrix. We then compute the eigenvectors of the Laplacian matrix, from which we extract the the k eigenvectors having the k smallest eigenvalues. These eigenvectors are concatenated in columns to form the matrix U. Finally, we cluster the matrix U along the rows into k clusters, using the k-means algorithm.

The main issue with the spectral clustering algorithm is that it needs to compute very large matrices when the dataset is large, which can be computationally expensive.

In the case of auroral image unsupervised classification, a variant of the spectral clustering algorithm has been used [4]. Instead of computing the adjacency matrix using Euclidean distance, an affinity matrix is used, which is computed by a Siamese Network. Then, the eigenvectors computation is done using the SpectralNet network [33].

The goal of SpectralNet is to approximate the spectral decomposition. It thus returns an approximation of the eigenvectors of the input vector cloud. The orthogonality constraint originally needed in spectral decomposition is only approximately verified. However, it provides a function which is an approximation of the eigenvectors of the graph Laplacian. This function can thus be applied to new points, and can be computed on large data sets, which is impossible with classic spectral clustering.

Hierarchical clustering

Hierarchical clustering [34] is based on the principle that clusters which are close to each other should be merged. At the beginning, all samples constitute their own single cluster. Then, for each pair of clusters is computed a score of distance between clusters. The pair with the lowest score is merged. The last two operations are repeated until the number of clusters corresponds to a user-defined constant.

There are multiple ways to define between-clusters distance:

- Complete linkage: Maximum distance among all pairs of observations between the two clusters.
- Average linkage: Average distance among all pairs of observations between the two clusters.
- Single linkage: Minimum distance among all pairs of observations between the two clusters.
- Ward's linkage: Difference between the sum of squared distances in merged cluster and the sum of squared distances of the two separate clusters. This score represents the cost of merging two clusters in terms of intra-cluster variance. This allows to make the growth of intra-cluster variance as small as possible between each iteration.

DBSCAN

DBSCAN [35] is a clustering algorithm which is based on the hypothesis that clusters are regions of high density. The main idea of this method is the concept of core sample, which is a sample located in a high-density region of the data set. There are two main parameters: $min\_samples$ and ϵ. A sample is considered a core sample when there are at least $min\_samples$ samples in a radius of ϵ around it. These other samples are defined as neighbors of the core sample.

A cluster is then defined in a recursive way by finding all neighbors of a core sample which are core samples themselves and repeating this search for all new core samples found. Samples which are not in the neighborhood of any core sample is considered an outlier and is not clustered.

The choice of the ϵ parameter is very important to tune, to find a compromise between considering all points as outliers and clustering all samples in the same clusters [36]. The DBSCAN method is interesting because contrary to methods such as K-means clustering, it does not require a predefined number of clusters and has no assumption of convex

clusters. The ϵ and *min_samples* parameters can be however hard to tune depending on the usage context [36].

3.7.2 Ensemble clustering

Every clustering method has its weaknesses and biases, and in some cases it is difficult to tell if one clustering result is better than another, as they can be equally plausible. A way of improving clustering quality is to combine multiple clustering results in order to compensate individual downsides.

These methods are grouped under the name "Ensemble Clustering"[37]. The main desired properties for ensemble clustering methods are the following:

- *Robustness*: The combination of the clustering methods must bring a higher performance than the single clustering results it is based on.
- *Consistency*: The computed result should not be too different from all the single clustering methods it originated from.
- *Novelty*: Ensemble clustering should provide a solution which would be impossible to reach with single clustering methods.
- *Stability*: The result should be less sensitive to noisy data and outliers.

The first step of an ensemble clustering method is the *Generation mechanism*. Its goal is to produce a set of clustering results based on the same dataset. Generation can be made by varying a clustering parameter such as the number of clusters, by using different clustering algorithms, or by clustering on different projections of the feature space. It is generally better to provide diverse solutions since it can bring more information for the next step, the *Consensus function*.

In this step, the goal is to find a new partition of the data which integrates the single clustering results generated in the preceding step. There is a wide variety of consensus functions, which can be divided into two categories of methods: objects co-occurrence and median partition.

In the first case we determine how many times each sample belongs to a cluster, or how many times two samples are clustered together. Then, through a voting mechanism, we form a new consensus partition.

In the second approach, we search the median partition of the clustering ensemble. In other words, we need to find the data partition which will maximize its similarity with all partitions of the clustering ensemble. This is an optimisation problem, which can be formalized as finding the partition P^* defined as:

$$P^* = \arg\max_{P \in \mathbf{P}} \sum_{j=1}^{m} S(P, P_j)$$

where $\mathbf{P}$ is the set of all possible partitions of the dataset, $(P_j)_{j=1...n}$ are all the partitions in the clustering ensemble, and S is a similarity measure between two partitions. Similarity measures correspond mainly to external cluster validity indices which will be described in detail in section 3.7.4.

One of the simplest ensemble clustering algorithm is the majority voting based on the co-association matrix. This method lies in the "objects co-occurrence" category. The first step is to generate the the set of partitions based on single clustering methods. In the

paper presenting this method [38], K-means clustering methods are used, with different initialisation seeds for the clusters centers. The number of clusters is chosen to be a high value, around $\sqrt{N}$, with N being the number of samples in the dataset.

The next step is to build the co-association matrix, which counts the number of times each pair of samples are clustered together in the same cluster. It can be formalized as:

$$CA_{ij} = \frac{1}{m} \sum_{t=1}^{m} \delta(P_t(x_i), P_t(x_j))$$

where $P_t(x_i)$ represents the label of the sample x_i in the partition P_t, and $\delta(a, b)$ is 1 if $a = b$ and 0 otherwise. This matrix can be viewed as a similarity measure between all objects of the dataset.

The next step is to use this co-association matrix to form the final clusters, by using a majority voting principle. For each pair of samples (x_i, x_j), if $CA_{ij} > 0.5$ then they belong to the same cluster. Each sample already connected to x_i or x_j are merged into the same cluster. For each sample which is not included in any cluster, we form a single element cluster.

3.7.3 Constrained clustering

Constraining the clustering process can be useful in the case where a small amount of data has already been labelled. We can use this information in order to guide the clustering method into creating more meaningful results. The constraints can be expressed in different ways. Must-link/cannot-link constraints provide information on which pairs of samples are in the same or different clusters, whereas relative comparisons constraints tell for a group of 3 samples x, y, z that x is closer to y than to z for instance. A small set of labelled samples can also be used as a starting point (also called a seed) for standard clustering methods.

One of the simplest constrained clustering method based on the K-means method is called COP K-Means [39]. The constraints used in COP K-means are expressed as two types of pairwise constraints:

- Must-link constraints which imply that some pairs of samples must end up in the same cluster.
- Cannot-link constraints implying that some pairs of samples must be separated into different clusters.

The COP K-means algorithm has the same base architecture as the regular K-means. However, instead of assigning each point to its closest cluster at each iteration, we assign it to the closest cluster which does not violate the constraints regarding this point. To check if a cluster C does not violate constraints for a data point d, we check the following rule: if every data point paired with d in a must-link constraint belongs to cluster C, and if every data point paired with d in a cannot-link constraint does not belong to C, then the constraints are not violated for the (d, C) association. In this case, the data point d can be added to the cluster C. Otherwise, we check the next closest cluster in the same way. We repeat the process until a cluster is found where no constraint is violated. If no such cluster exists, the algorithm fails. The updating of the cluster centroids is then computed in the same way as the regular K-means algorithm.

3.7.4 Validity scores

Assessing the quality of the clustering results is critical in unsupervised classification contexts. Because we do not have access to a ground truth, traditional performance measures are not usable. Therefore, specific clustering metrics have been developed to try to assess the clustering performance. These can be sorted into 3 categories: External validation indices which use comparison with a reference, Internal validation indices which use information intrinsic to the data, and stability metrics which measure the sensitivity of the results to small changes in the initial conditions.

External validation indices

External validation indices allow to compare 2 assignments of clustering algorithms, giving an indication on how much they agree, while ignoring cluster permutations. They can be seen as consensus measures between two data partitions. When using them against ground truth labels, they can be seen as a performance metric. Such methods are not usable in real-world applications, but are still useful to compare two different clustering results, for stability analysis for instance.

Two of the most used external validation metrics are the Adjusted Mutual Information score and the Adjusted Rand index.

The mathematical formulation of the Mutual Information (MI) score is the following.

Given 2 cluster label assignments U and V of the same N samples, the entropy of the assignment U is defined as:

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$$

where $|U|$ is the number of clusters in U, and $P(i) = \frac{|U_i|}{N}$ is the probability that a randomly taken sample belongs to the i^{th} cluster of the partition U. The entropy describes the amount of uncertainty for a given partition set. Similarly, we have:

$$H(V) = -\sum_{j=1}^{|V|} P'(j)\log(P'(j))$$

where $|V|$ is the number of clusters in V, and $P'(j) = \frac{|V_j|}{N}$.

The Mutual information between U and V is then defined as:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

where $P(i,j) = \frac{|U_i \cap V_j|}{N}$ is the probability that a randomly chosen sample belongs to both clusters U_i and V_j.

We can then define the Normalized Mutual Information (NMI):

$$NMI(U,V) = \frac{MI(U,V)}{mean(H(U),H(V))}$$

This score is not adjusted for chance. This means that its value will increase with the number of clusters, regardless of the actual consensus between the two measured clustering algorithms. Therefore, an adjusted MI score can be defined, using the expected value for the Mutual Information, noted $\mathbf{E}(MI)$. Let $a_i = |U_i|$ the number of elements in the i^{th} cluster of U, and $b_i = |V_j|$.

$$\mathbf{E}(MI) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \sum_{n_{ij}=(a_i+b_j+N)^+}^{min(a_i,b_j)} \frac{n_{ij}}{N} \log(\frac{Nn_{ij}}{a_i b_j}) \frac{a_i! b_j! (N-a_i)! (N-b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!}$$

where $(a_i + b_j - N)^+$ denotes $\max(1, a_i + b_j - N)$. Thus, the Adjusted Mutual Information (AMI) can be expressed as:

$$AMI(U,V) = \frac{MI(U,V) - \mathbf{E}(MI(U,V))}{mean(H(U), H(V)) - \mathbf{E}(MI(U,V))}$$

Another External validation index is the Adjusted Rand score. Let U,V be two cluster assignments of the same dataset with N samples. We defined the following:

- a, the number of pairs of samples which are in the same cluster in U and in the same cluster in V.
- b, the number of pairs of samples which are in different clusters in U and in different clusters in V.
- c, the number of pairs of samples which are in the same cluster in U and in different clusters in V.
- d, the number of pairs of samples which are in different clusters in U and in the same cluster in V.

The Rand index is defined as:

$$R = \frac{a+b}{a+b+c+d}$$

It represents the frequency of occurrence of agreements over all pairs in the dataset.

The Adjusted Rand Index (ARI) accounts for chance and is defined as:

$$ARI = \frac{a - \frac{(a+c)(a+d)}{a+b+c+d}}{\frac{(c+d+2a)}{2} - \frac{(a+c)(a+d)}{a+b+c+d}}$$

Internal validation indices

Internal validation indices aim to assess the quality of the clustering structure without any external reference. They only use information which is intrinsic to the data, and in particular the compactness of all clusters as well as the separability of the clusters. Such indices are practical, however they are meaningful only when the found clusters have a convex shape. There are three main internal validation indices: the Silhouette coefficient, the Calinski-Harabasz index and the Davies-Bouldin index.

The Silhouette coefficient [40] is first defined for each sample using the following quantities:

- a, the mean distance between a sample and all other samples in the same cluster.

– b, the mean distance between a sample and all other samples in the nearest cluster.

The Silhouette coefficient is then defined as :

$$S = \frac{b - a}{max(a, b)}$$

In order to characterize the whole dataset, we take the mean value of the silhouette score on all samples. A value close to 1 means that the clusters are well separated, whereas a value close to 0 indicates a poor clustering result.

The Calinski-Harabasz index [41], also called Variance Ratio Criterion can be also used as an internal validation index. It is defined as the ratio of the between-clusters dispersion mean and the within-cluster dispersion. We first need to defined the between-clusters dispersion matrix B_k and the within-cluster dispersion matrix W_k, which depend on the number of clusters k:

$$B_k = \sum_{i=1}^{k} n_i (c_i - c_D)(c_i - c_D)^T$$

$$W_k = \sum_{i=1}^{k} \sum_{x \in G_i} (x - c_i)(x - c_i)^T$$

Where n_i is the number of samples in the cluster i, c_i is the center of the cluster i, c_D is the center of the whole data set and G_i is the set of samples in the cluster i. The Calinski-Harabasz index is then defined as:

$$CH = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_D - k}{k - 1}$$

Where tr is the matrix trace operator and n_D the total number of samples in the data set. This score has a higher value if the clusters are dense and well separated.

Another internal validation index is the Davies-Bouldin index [42]. It is defined as the average similarity between each cluster and its most similar one. In this case, similarity is expressed by the matrix R_{ij} defined by:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Where s_i (resp. s_j) is the mean distance between each sample of cluster i (or j) and the centroid of this cluster, and d_{ij} is the distance between centroids of cluster i and cluster j. The Davies-Bouldin index is then defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

When this score approaches zero, this means the clusters are dense and well-separated.

Clustering stability analysis

The goal of analysing a clustering method's stability is to assess the similarity between clusters obtained from slightly different initial conditions. These can be obtained for instance by taking independent and identically distributed samplings of the same data set, by applying random noise to the data vectors, or by using representations based on

different reduction methods. A stable algorithm is expected not to change greatly when training from slightly different initial conditions or with partial information.

To measure the similarity between the different results, we can use for instance external validation indices such as the Adjusted Mutual Information or the Adjusted Rand index. By studying the distribution of the similarities between all pairs of partitions obtained through different initial conditions, we can have an information on the stability of the method [43]. This allows to know whether the clusters are meaningful or are only random artifacts from the algorithm.

It is most pertinent to use stability criteria for detecting a highly unstable clustering method, thus distrusting its results. However, if a clustering method is more stable than another, it does not mean it has better clusters because many other factors can influence the stability of a clustering [44].

3.8 Results of the state of the art

We report here some of the recent results in the literature related to auroral classification. Because the application context is different for each research paper, we cannot rigorously compare their results quantitatively. Their conclusions can however help us decide which methods seem promising in the auroral classification context.

In [5], 5 deep learning models (VGG-16, VGG-19, AlexNet, ResNet-18 and ResNet-50) were trained from scratch on labelled all-sky color images from the Kiruna observatory. They were compared to a KNN classifier and a SVM classifier trained on features extracted by a histogram of oriented gradients (HOG). The best results were obtained using the ResNet-50 architecture. The authors noted however that some confusion remained, especially between the colored and Discrete classes. Some other classes such as Arcs or Patchy achieved good precision, but could be subdivided into more classes, such as single/multiple arcs. More generally, deep learning based methods seemed to outperform standard machine learning methods, especially for identifying complex auroral structures.

In [7], 80 pretrained neural networks were tested for supervised auroral image classification on the OATH data set. These networks were trained on the ImageNet data set and were only used a feature extractor. A simple classifier such as a Support Vector Machine was used to sort images into the auroral classes as well as the NoAurora, Cloudy or Moon labels. Their conclusion was that the difference in performance between the neural networks was very small, therefore they could be considered as equivalent in this task. The margin of improvement seemed to lie more on the type of the classifier.

The deep convolutional autoencoder implemented in [4] was applied to the filtered grayscale images from the Arctic Yellow River station in order to extract the auroral image features without using manual labels. Then an unsupervised classification was made using the deep learning version of the spectral clustering algorithm. With tested values of the number of clusters k from 2 to 10, the best values for the Silhouette, Davies-Bouldin and Calinski-Harabasz indices were found for $k = 2$. The first cluster seemed to contain distinct shape features, regrouping arcs, patches and spot-like auroras. The overall intensity of the images in the first cluster was higher than in the second cluster. The second cluster mainly contained corona auroras with rich texture details. The low number of clusters however did not allow to have a fine classification of the auroral structures.

The SimCLR unsupervised deep learning method was used in [10] with a ResNet18 model with random initial weights as its encoding part. The projection head was composed of two dense neural layers with a ReLU activation in between, without biases. The temperature parameter was equal to 0.5. This method was applied to the OSLO Auroral THEMIS data set. The encoded output of the SimCLR network was then clustered using K-means clustering. The Silhouette score was computed for values of k between 3 and 15. The maximum score (0.212) was obtained for $k = 12$. When the 6 manual labels from the OATH data set were used for clustering, the silhouette score was lower (0.011). Classes such as "Cloudy" or "Moon" aligned well with the computed classes. However, others like the "Discrete" class were split into different clusters. Samples among the learned clusters had high qualitative similarities. The researchers claimed that this approach gave a better categorization than the initial labelling, with more categories.

4 Implementation details

4.1 Overview

In this project, our goal is to provide most precise and meaningful groupings of color auroral images based on their morphology. Therefore, in our implementation we will first need to create an image data set which is sufficiently exploitable, through manual labelling and automated pre-processing. After these steps, 6 feature extractors based on unsupervised machine learning are used to encode the data set into 6 different feature spaces. Then each of them undergoes 5 types of dimension reduction operations, one of them being the absence of reduction. The output number of dimensions is determined automatically for each feature space. Each of the 30 reduced data sets is then used to train up to 4 clustering algorithms, which will provide a cluster label for each sample of the data set. Two different numbers of clusters have been used. We then use qualitative and quantitative criteria to sort the 113 results listed in Table 4.2. Because each combination of feature extractor, reducer and clustering method has its limitations, we want to combine some of the results using an ensemble clustering principle. We expect the combined results to be better than the sum of its parts. The processing chain is synthesized in Figure 4.1.

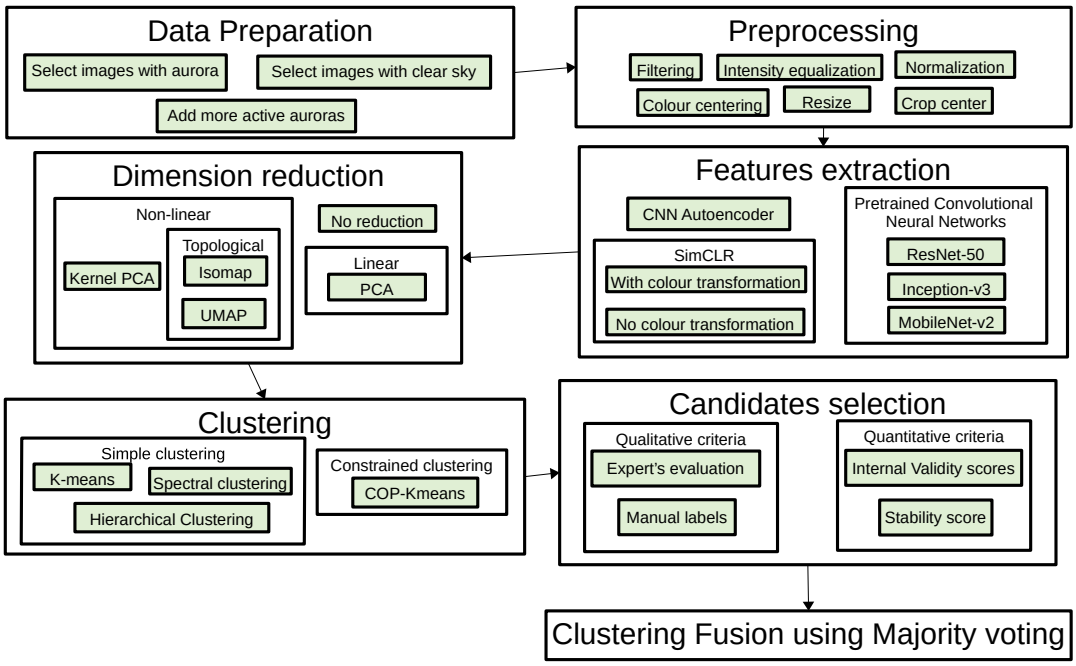


FIGURE 4.1: Synthetic diagram of the whole processing chain.

4.2 Data preparation

Before any processing, we need to prepare the data in order to have a "clean" dataset, which only contains clearly identifiable auroras.

To do so, we have around 37000 images from 4 months of acquisition from winter 2019-2020 as well as the months of January and February 2019 which have been manually classified in two categories depending on whether they contain an aurora or not. We will then only use the 10,916 images labelled as containing auroras.

The main sources of image perturbation are the presence of clouds and the Moon, especially around full-moon periods. The presence of the Moon however does not entirely modify the appearance of auroras, and in a number of cases the images can still be exploitable. It will however introduce a bias, as most image feature extractors are sensible to high luminosity changes.

Contrarily, the presence of clouds can completely change the appearance of auroras in terms of brightness, shape and texture. Instead of a bias, clouds introduce factors of confusion between auroral shapes. Therefore, we will exclude all auroral images which are not manually labelled as containing clouds. As a result, only 9,061 images are left.

This image set has a time resolution of 6 minutes. This allows to accurately follow the evolution of auroras during quiet periods as they evolve slowly. However, more active auroras evolve at much smaller timescales, and present most of the diversity in terms of morphology. Therefore, a fixed time interval creates an imbalance towards quieter auroras and fails to capture most auroral forms of interest. To solve this issue, 19 time periods featuring high auroral activity have been selected. In these periods, the time resolution has been reduced to 24 seconds. In both quiet and active periods, the time resolution has been selected so that two consecutive images are not too similar. This avoids possible biases in further processing steps. In the end, the data set which will be used in the next steps contains 14,170 images.

The images in this data set are already reduced in size (480x480) compared to the raw images (2400x2400). In addition, a text row at the bottom of the image indicating the acquisition conditions is present, thus forming images with a size of 510x480.b

In order to have a reference for assessing the pertinence of the clustering methods we have a small amount of labelled images (Figure 4.2). The classes found here do not aim to be exhaustive, but rather give reference samples for some typical structures which can be found in the data set. The classes are the following:

- *SingleArc* (12 elements): The images in this class contain only one auroral structure in the shape of an arc, which can be varying in brightness, color and location in the sky.
- *MultipleArc* (13 elements): This class contain images with two or more auroral arc-shaped structures organized in parallel bands.
- *LargeVortex* (12 elements): This class contains auroral structures which are very bright and take up a major fraction of the sky with complex swirling shapes. They are mainly observed during active phases and evolve quickly over time.
- *Diffuse* (8 elements): *Diffuse* auroras are very dim and cover a major part of the sky like a transparent veil.

– *Patchy* (14 elements): *Patchy* auroras have the same brightness as *Diffuse* auroras but have a more irregular texture with many small cloud-like structures.

– *Corona* (16 elements): Auroras of this class have a structure with bright rays extending outwards from the zenith, with often reddish hues. This corresponds to when the aurora is located above the observation location.

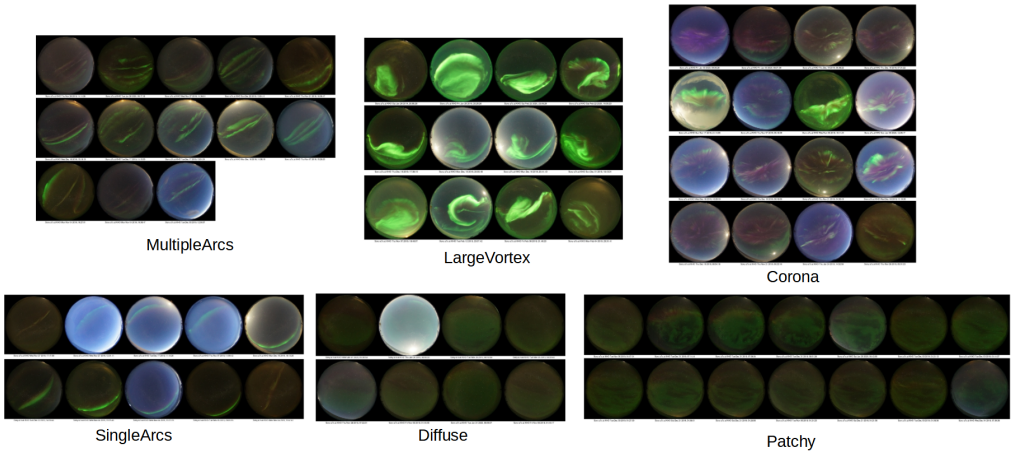


FIGURE 4.2: All samples from each reference class labelled for this project.

4.3 Preprocessing

The preprocessing step in our context will consist not only in removing unnecessary information and adapting the format of the input images, but also in enhancing and standardising them, to facilitate the unsupervised classification and avoid biases as much as possible (see Figure 4.3).

We first remove the bottom text of the image to form a centered square image of size 480x480.

Then we reject images which are too dim. Indeed, some auroras are almost imperceptible because they are not bright enough or they are at very low elevation angles (see Figure **??**). To measure the overall brightness of color images, we first use the HSV (Hue,Saturation, Value) color system to isolate the Value channel, correlated with image brightness. If the 90^{th} percentile of the Value channel is under the threshold of 50 out of 255, we consider the image as containing no exploitable auroral features, thus excluding it. As a result, 3862 images are excluded, leaving a data set of 10308 images in total.

In the next steps we will enhance and standardize the image, in terms of brightness as well as color. To work with brightness and color independently, we use the L*a*b* color space [45]. Its first channel represents the lightness of the color ($L^* = 0$ is black whereas $L^* = 255$ is white), the second channel represents the color's position between green and red and the third is the position between blue and yellow. Before any further processing, we want to ignore the black corners due to the fish-eye lens. To do so, we set all pixels with L* value under 0.01 to the 'Not a Number' value. All further processing steps will then ignore such values.

Next, in the same way as most preprocessing chains for auroral images [8][7][10], we clip the pixel values between the 10^{th} and 90^{th} percentile in the L* channel.

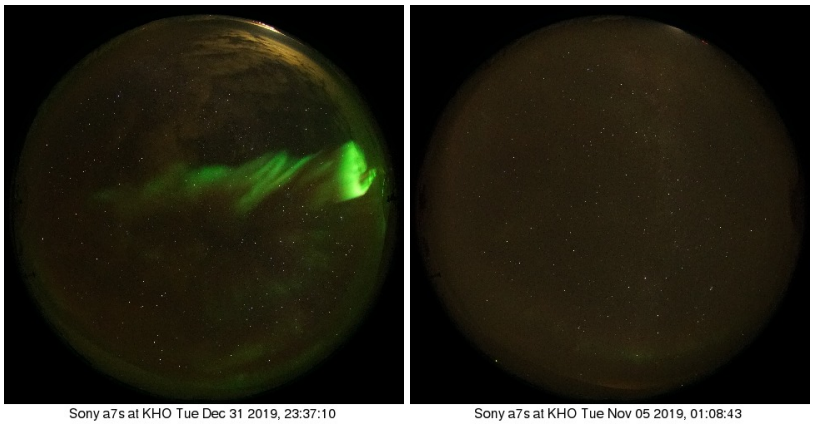


FIGURE 4.3: Examples of two raw images from our data set. On the left is a sufficiently bright image. On the right, the image is too dim to be retained for further processing.

Because of the transparent nature of auroras and the diverse sky lighting conditions, an important bias in terms of color is present. Indeed, the intrinsic color of auroras are often shifted due to the color of the background sky. The resulting risk is to classify images according to the background sky lighting conditions only. To avoid this bias, we aim to shift the color channels so that every image has the same centered color bias. To do so, we shift all values of channels a* and b* by the difference between the median value and 128, which is the middle of the whole color range between 0 and 255. Then, we clip values which end up below 0 or above 255. This way, the color channels of every image has approximately the same centered median value of 128.

Next, we crop the center of the image, leading to a 400x400 image. This aims to avoid biases from the round borders and black corners. At this point, no NaN pixel value remains. In addition, because we will focus on auroral morphology, auroras which are located at low elevation angles (thus on the border of the image disk) are not usable, as they are heavily distorted. Therefore, focusing on the center of the image is justified in this context.

Because a large number of images contain faint auroras which would be difficult to describe in terms of morphology, we add a contrast enhancement step. It consists of a histogram equalisation of the L* channel, followed by the same median centering step which was used on the a* and b* channels.

Then, we use a 5x5 median filter to remove the stars from the background sky. The image is then resized to a 224x224 image with normalized RGB pixel values between 0 and 1. This format corresponds to most of the expected inputs of the feature extractors used in later steps such as ResNet-50 for instance. Extractors expecting another type of format will have their input reformatted on the fly before processing. The result of the preprocessing step can be seen in Figure 4.4.

4.4 Features extraction

In the feature extraction step, we choose to use methods which have already been proven to perform well on auroral images classification and that can be applied in the context of an unsupervised classification. The goal is also to try methods which have diverse

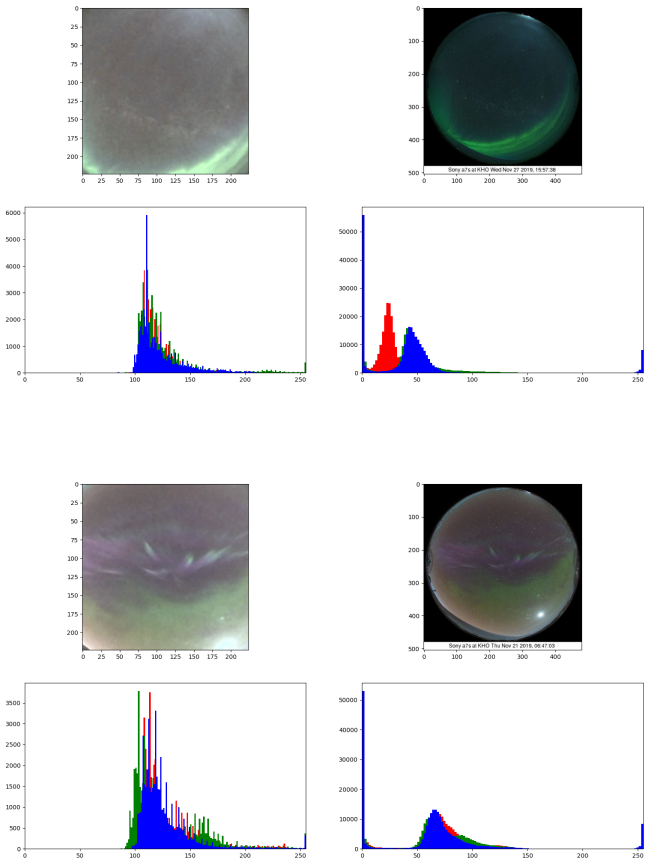


FIGURE 4.4: Example of two raw (right) and preprocessed (left) images along with their RGB histograms. We can see the effect of centering and equalizing on the color histograms.

representation mechanisms, in order to have complementary information on the data and avoid redundant representations.

First, we choose 3 pretrained deep convolutional neural networks: ResNet-50, Inception-v3, MobileNet-v2. These networks have been trained on the ImageNet data set[11], and are ranked among the best performing networks on supervised classification. They also have a relatively high computing speed (respectively 58.2, 42.2, 25.9 ms per inference step compared to 69.5 ms for a standard VGG16 architecture, on Python implementations [46]). The three architectures are also diverse in their layer organisation, which could mean that they can capture different characteristics in the images.

To use these pretrained models, we remove the last classification layer and instead introduce a global average pooling after the last convolution layer. This produces a feature vector of size 2048 for ResNet-50 and Inception-v3 and of size 1280 for MobileNet-v2. These networks cannot be trained and are only used in inference mode.

The second type of feature extractor is the convolutional Autoencoder, which has already been used for unsupervised classification of gray-scale auroral images [4]. The version implemented here is however simpler and more light-weight than in the state-of-the-art version. The encoding part is composed of 4 convolutional layers with a stride of 2, thus reducing the size of the images by a factor of 2 per layer. Each layer has a set of

88 convolution filters of size 3x3. The output of the last layer is then flattened to give a feature vector. The input image size is $24 \times 224 \times 3$, therefore the feature vector has a size of $14 \times 14 \times 88 = 17248$.

The decoding part first reshapes the feature vector into 88 channels of size 14x14. Then in a symmetrical way, there are 4 Transpose convolution layers with 88 filters each which increase the size of the image by a factor of 2 per layer. All filters have a 3x3 size and every layer has a 'ReLU' activation function. As a final step, the output is passed into a convolution layer with 3 filters, with a sigmoid activation function in order to provide a 3-channel output image.

The number of filters and the number of layers have been determined through trial and error to minimize the reconstruction loss with the constraint of having at least 3 layers in order to have a spatially compressed feature vector. The reconstruction loss is obtained trough the mean squared error between the original and reconstructed image. The data set is randomly split into 90% of training images and 10% of validation images, and we use the Adam optimizer with a learning rate of 0.002 to train over 10 epochs. The obtained reconstruction loss is 0.00023 on the training set and 0.00024 in the validation set.

The last type of implemented feature extractor is the SimCLR model. The goal here was to replicate as much as possible the implementation of [10]. We use here a ResNet-50 architecture as the main feature extractor. The projection head is composed of two neural layers of 256 and 128 neurons. With a 'ReLU' activation layer in-between.

Two sets of random image transformation have been used:

- The first is composed of horizontal flipping with a 50% probability, a random cropping with 50% probability with a size ranging from 30% to 90% of the initial size and a random position, followed by a resizing operation to the initial size. The resulting model is called SimCLR-NCT.
- The second includes all cited operations and adds color transformations: color drop and color jitter, with respectively 20% and 80% probabilities. color drop consists in replacing each color channel by the gray-scale image. color jitter is a combination of brightness, contrast, saturation and hue random adjustments. It also adds a vertical flipping of the image with 50% probability. Even though these transformations were not present in the past implementation, they were presented as improving performance of the network in a general context [22].The resulting model is called SimCLR-CT.

Both versions of the SimCLR networks are trained using the Adam optimizer with a learning rate of 0.001. The temperature parameter used for computing the loss is set to 0.5 and the batch size on which it is computed is 64. After training on 10 epochs with 90% training data and 10% validation data, the validation loss reached a value of 6.37 for the version of the network without color transformation. Even though the loss value is high, the learned feature extractor still gives meaningful results, as we will see in the next steps.

In the same conditions, the ResNet-50 has been replaced with an Inception-v3 architecture. The resulting validation loss reached 6.44 which is not an improvement compared to the initial setup. In the same way, changing the temperature parameter in the range of 0.001 to 2 does not improve the performance. The original setup is thus kept for the next processing steps.

For each type of feature extractor, the feature vectors data is stored in memory in order to avoid computing again the model inferences.

4.5 Dimension reduction

Each encoded dataset is processed using 5 types of dimension reduction.

- The first is only the absence of dimension reduction. This will allow to have a reference for comparing the other dimension reduction methods. This can also tell if dimension reduction is a necessary step to include.
- Then a linear and kernel PCA are used. The kernel is a radial basis function[1].
- An Isomap reduction method is also implemented. The main parameter to choose here is the number of neighbors for the graph computation. To see its influence on the data representation, we compute the reduced features of data extracted by the SimCLR model with no color transformation, with a final dimension of 3 and we change the number of neighbors from 2 to 50 (Figure 4.5). Starting from a value of 5, reduction artifacts disappear and the global structure keeps approximately the same shape. The number of neighbors is thus set to 5, as minimising this value allows to have lower computing times.
- A final dimension reducer is implemented which is the UMAP method. In this case two parameters are important to set: the number of neighbors in the graph computation as well as the minimum distance between points in the final representation. In theory, we can say that a small *n_neighbors* value will tend to preserve more the local structure of the data. However it can create artificial clusters which are only a result of noise in the original data. Contrarily, a high *n_neighbors* value will allow to take the overall structure into account but also creates the risk of losing some local structure information. The goal here is to avoid false clusters while keeping enough local information to have meaningful clusters in the final representation.

  For the *min_dist* parameter, because we are in a clustering context, we want to have a high density in the final embedding, therefore its value should be set to 0 [47].

  To see the influence of the *n_neighbors* parameter, we compute the reduced UMAP features in the same context as with Isomap, with values of *n_neighbors* from 3 to 30, and $min\_dist = 0$ (Figure 4.6). Starting from a value of 10, the global structure stays approximately the same but the density decreases slightly. To avoid any remaining artificial clusters while keeping a high density, we choose $n\_neighbors = 20$.

For all dimension reduction methods the output number of dimensions is set according to the following rule. For each type of feature extractor, we compute the PCA reduction of the feature data with the output number of dimension set to 100. Each time, we compute the amount of variance explained by each dimension. The minimum number of dimensions which provides a cumulative explained variance ratio of at least 80% is chosen as the value for further processing (see Table 4.1).

This rule allows to have an approximation of the underlying number of dimensions in each feature data set and then adapt the final embedded dimension number to keep most of the information while sufficiently reducing the size of the feature vectors.

[1] The radial basis function kernel is expressed for two points x_1, x_2 as follows: $K(x_1, x_2) = \exp(-\frac{||x_1 - x_2||^2}{2\sigma^2})$

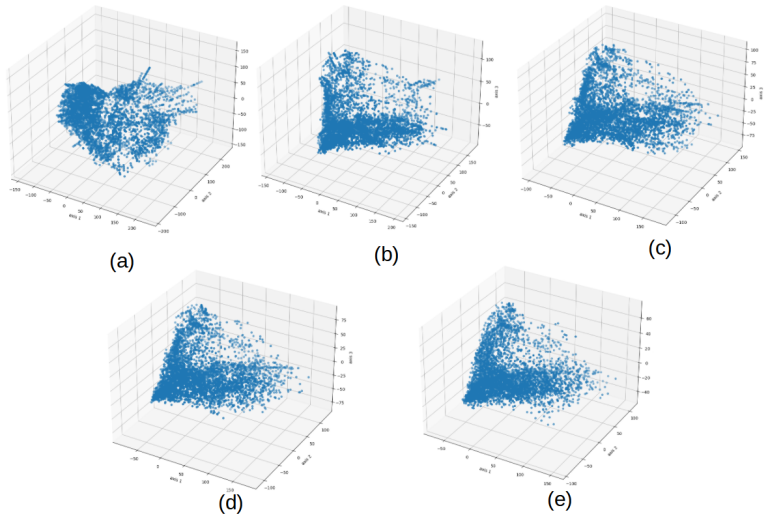


FIGURE 4.5: 3-dimensional point cloud representation of the feature data extracted by the SimCLR-NCT method and reduced by Isomap, with varying *n_neighbors* values: (a) $n\_neighbors = 2$, (b) $n\_neighbors = 3$, (c) $n\_neighbors = 4$, (d) $n\_neighbors = 5$, (e) $n\_neighbors = 10$

Feature extractor	Computed number of dimensions
ResNet-50	11
MobileNet-v2	13
Autoencoder	27
SimCLR-NCT	3
SimCLR-CT	5
Inception-v3	8

TABLE 4.1: Chosen number of dimensions for each feature extractor, using the criterion of keeping 80% of the explained variance with standard PCA.

4.6 Clustering

In total, four types of clustering methods have been implemented: K-Means, Spectral Clustering, Hierarchical Clustering and COP K-means.

Every clustering method chosen here needs to have its number of clusters K defined. To do so, we could use internal validation scores to determine the best number of clusters for each combination of extractor and reducer. However, in our case the data is noisy and does not contain clear and distinct clusters but rather continuous changes between regions. As a result, internal scores have overall low values (Silhouette scores are always lower than 0.5 and the Davies-Bouldin indices are always higher than 0.7) and are therefore highly noisy. It is thus difficult to automatically find an optimal value in such conditions. In addition, for a same combination the optimal K value often differs greatly depending on the selected score (see Table 4.7). We also cannot use a stability criterion for searching K as it has been proved to be not pertinent for this task [44].

Thus, we are forced to choose a value of K manually. Because we will try to combine different clustering results in the next steps using a majority voting principle, we can follow the rule of thumb which was presented along with this method [38]. It states that best performance is achieved when the number of clusters of the single clustering methods is approximately equal to $\sqrt{N}$ with N the total number of samples in the data set. We

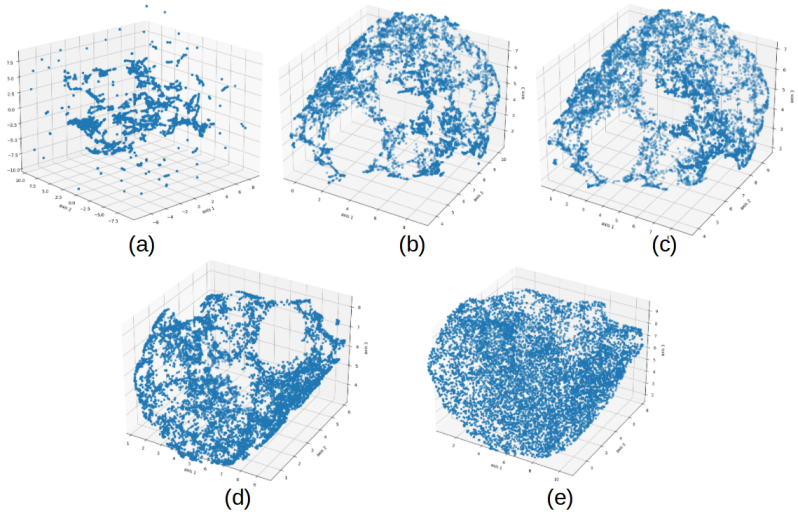


FIGURE 4.6: 3-dimensional point cloud representation of the feature data extracted by the SimCLR-NCT method and reduced by UMAP, with varying *n_neighbors* and *min_dist* values: (a) $n\_neighbors = 3$, (b) $n\_neighbors = 10$, (c) $n\_neighbors = 20$, (d) $n\_neighbors = 30$, (e) $n\_neighbors = 30$. $min\_dist = 0$ for plots (a) to (d) and $min\_dist = 0.5$ for (e)

therefore first set $K = 100$ as there are around 10,000 samples in our data set. However because this number of clusters is very high compared to any previous structure classifications of the aurora, we will also compute the results for $K = 30$, which is more similar to the final number of clusters we will want to have. In general, we want to compute more clusters than manually labelled classes, because we know that there exists more types of auroras than what we are able to classify manually.

The K-means implementation used here corresponds to the *K-means++* version described in section 3.7.1. The COP K-means method uses the same *K-means++* basis. The must-link and cannot-link constraints are computed from the 6 labelled classes available, creating a set of 67 must-link and 5 cannot-link constraints. We implement here only the original version of Spectral Clustering for simplicity. The hierarchical clustering is implemented using the Ward's linkage.

K-means, Spectral and Hierarchical clustering methods follow the scikit-learn implementation [48], whereas the COP K-means implementation is inspired by [49].

To generalize the results of each clustering to new data, a K-neighbors classifier is trained on the labels found with the clustering method. We choose in all cases a number of neighbors equal to 5.

All computed combinations are listed in Table 4.2.

4.7 Selection of clustering results

Once all considered combinations are computed, we have to asses the performance of each clustering result in order to find the best elements. To do so, we have the 3 internal validity indices (Silhouette, Davies-Bouldin and Calinski-Harabasz indices) and the stability measure.

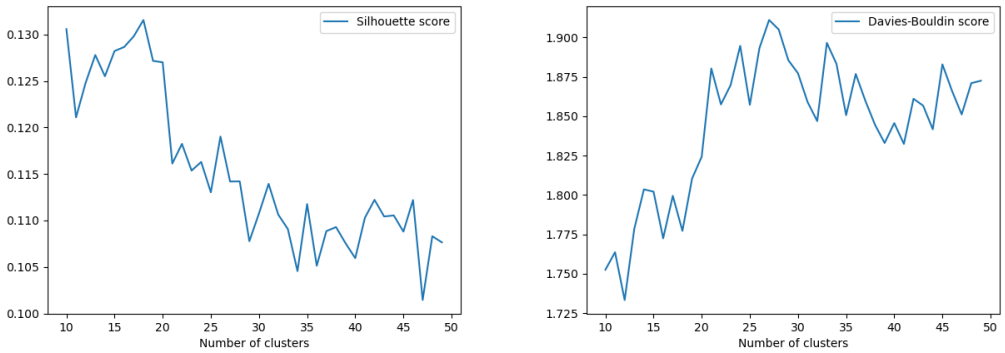


FIGURE 4.7: Example of computed Silhouette and Davies-Bouldin scores according to the number of clusters on feature data extracted by the autoencoder and reduced by a Kernel PCA. The maximum value for the Silhouette score is reached for $K = 18$ whereas the minimum Davies-Bouldin score is reached for $K = 12$

To compute the stability measure we randomly choose 10% of data points and compute the cluster labels on this subset. Then, using the k-nearest neighbors classifier (k=5) we extend the labels to the whole data set. Then we repeat this process by taking another 10 % of the dataset for training the clustering algorithm. Then we compute the similarity between each pair of results using the adjusted Rand index (ARI) and the adjusted mutual information (AMI). The stability measure is then the average between the mean ARI over all pairs of results and the mean AMI over all pairs of results.

We then need to make a ranking of the clustering results based on the 4 quantitative criteria. Because they are not directly comparable, we cannot combine them directly to compute the average score for instance. Instead, we sort the results according to each score, and then sum the 4 ranks of each solution. Sorting the results according to the sum of ranks gives the final ranking.

We will also verify potentially interesting results to see if the clusters are pertinent. For the verification, we will use a visualisation of 20 random samples from each cluster to see the intra-cluster homogeneity as well as a visualisation of the 5 nearest neighbors of each manually labelled sample along with their cluster number to see the pertinence of the clusters in terms of physical meaning.

4.8 Fusion of clustering results

In order to find more complex cluster representations which for instance do not involve convex clusters only, we combine the most promising results using a method based on the majority voting with co-association matrix algorithm [38].

The main difference with the original algorithm is that the input results originate from different feature extractors, dimension reducers and clustering methods. In addition, the majority threshold above which two samples are merged in the same cluster is not fixed but rather can be tuned between 0 and 1. Furthermore, instead of leaving non-merged samples in their own single cluster we choose to group them in one 'outliers' cluster, of label '-1'. We can also choose a minimal cluster size under which a cluster is merged with the 'outliers' cluster. This allows to have a cluster containing all the samples which are hard to classify.

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
Autoencoder	Identity	Hierarchical	100
		Kmeans	100
		Spectral	100
	Isomap	Hierarchical	100
		Kmeans	100
		Spectral	100
	Linear PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
Inception_v3	Identity	Hierarchical	100
		Kmeans	100
		Spectral	100
	Isomap	Hierarchical	100
		Kmeans	100
		Spectral	100
		COP Kmeans	100
	Linear PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
ResNet-50	Identity	Hierarchical	100
		Kmeans	100
		Spectral	100
	Isomap	Hierarchical	100
		Kmeans	100
		Spectral	100
		COP Kmeans	100
	Linear PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
MobileNet_v2	Identity	Hierarchical	100
		Kmeans	100
		Spectral	100
	Isomap	Hierarchical	100
		Kmeans	100
		Spectral	100
		COP Kmeans	100
	Linear PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
		Spectral	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
SimCLR_CT	Identity	Hierarchical	100
		Kmeans	100
	Isomap	Hierarchical	100
		Kmeans	100
	Linear PCA	Hierarchical	100
		Kmeans	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

Features Extractor	Dimension Reduction	Clustering method	Number of clusters
SimCLR_NCT	Identity	Hierarchical	100
		Kmeans	100
	Isomap	Hierarchical	100
		Kmeans	100
	Linear PCA	Hierarchical	100
		Kmeans	100
	Kernel PCA	Hierarchical	100
		Kmeans	100
	UMAP	Hierarchical	100
			30
		Kmeans	100
			30
		Spectral	100
			30
		COP Kmeans	100
			30

TABLE 4.2: Listing of all the computed combinations of feature extractor, dimension reduction, clustering method and number of clusters

The goal is then to find the 'best' combination of single results as well as the optimal values for the majority threshold and minimum cluster size. We first need to define an 'ideal' solution which could allow to create a loss function and thus optimize the parameters. We define it in terms of cluster size and number of clusters. We want ideally to have n_c clusters which are evenly distributed in size, with as few outlier samples as possible. The loss score is expressed as:

$$L = \frac{1}{n_c} \sum_{k=0}^{N} |n_k - n_0^k| + \frac{n_e}{n_c} \qquad with\ n_0^k = \frac{N}{n_c}\ \ if\ k < n_c,\ else\ n_0^k = 0$$

with N the total number of samples, n_k the number of samples in the cluster k, and n_e the number of outlier samples (see Figure 4.8). In our case, we choose $n_c = 25$. Even though this value is arbitrary, it allows to have a reference which penalizes having too much ($k > 100$) or too few ($k < 10$) clusters.

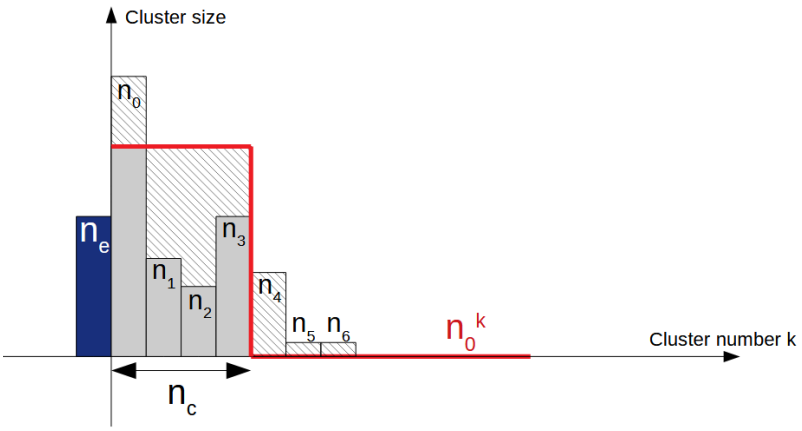


FIGURE 4.8: Schematic example of the loss function for selecting the fusion parameters.

5 Results and interpretation

5.1 Single results ranking

We have ranked all of the 113 combinations according to the different scores (see Figure 5.1 for the Silhouette score, and Annex A for the other scores).

We can see that the results which use the UMAP dimension reduction method are all among the best performing results. It is interesting to note that there is very little change of scores between a combination with 100 clusters and a combination with 30 clusters. Among the 30 best results, the scores vary only a little, which means they cannot be easily sorted and are equally pertinent candidates for a good clustering solution.

The 20 best results have been manually validated by visual inspection of cluster samples as well as the neighbors of the reference samples. All of them have a form of homogeneity among the samples of a same cluster. Even though each result has its weaknesses and strengths, they are all suitable for us to use in the fusion step.

5.2 Clustering fusion results

We first combined all the results without distinction between the extraction methods, first for $K = 100$ and then $K = 30$. We compute the value of the loss function according to 3 parameters: the number of clustering results to combine (we take the n highest ranked results) in a range of 2 to 30 with a step of 1, the majority threshold in a range of 0.4 to 0.9 with a step size of $\frac{1}{n\_results}$ and the minimum accepted cluster size in a range of 2 to 50 with a step size of 4. None of the computed solutions seem to be suitable. Indeed, the loss value always exceeds 550 for $K = 100$ and 650 for $K = 30$, which means that either the number of clusters is too high or too low, or there is too much excluded samples. We make the hypothesis here that the low performance is due to the important diversity in the data representations created by the different feature extraction methods.

We will then only combine results which originate from the same feature extractor, with no distinction for the number of clusters. The same conditions for the parameter space are used. The obtained fusion results are more encouraging. Indeed, for all extractors the minimal loss values is inferior to 200 (see Table 5.1) which corresponds to suitable values for the number of clusters and number of outlier samples.

5.2.1 Solution with SimCLR-NCT

We will present here the results obtained by combining 4 partitions of the data extracted by the SimCLR-NCT extractor. The majority threshold used is equal to 0.5 and the minimum cluster size is set to 46.

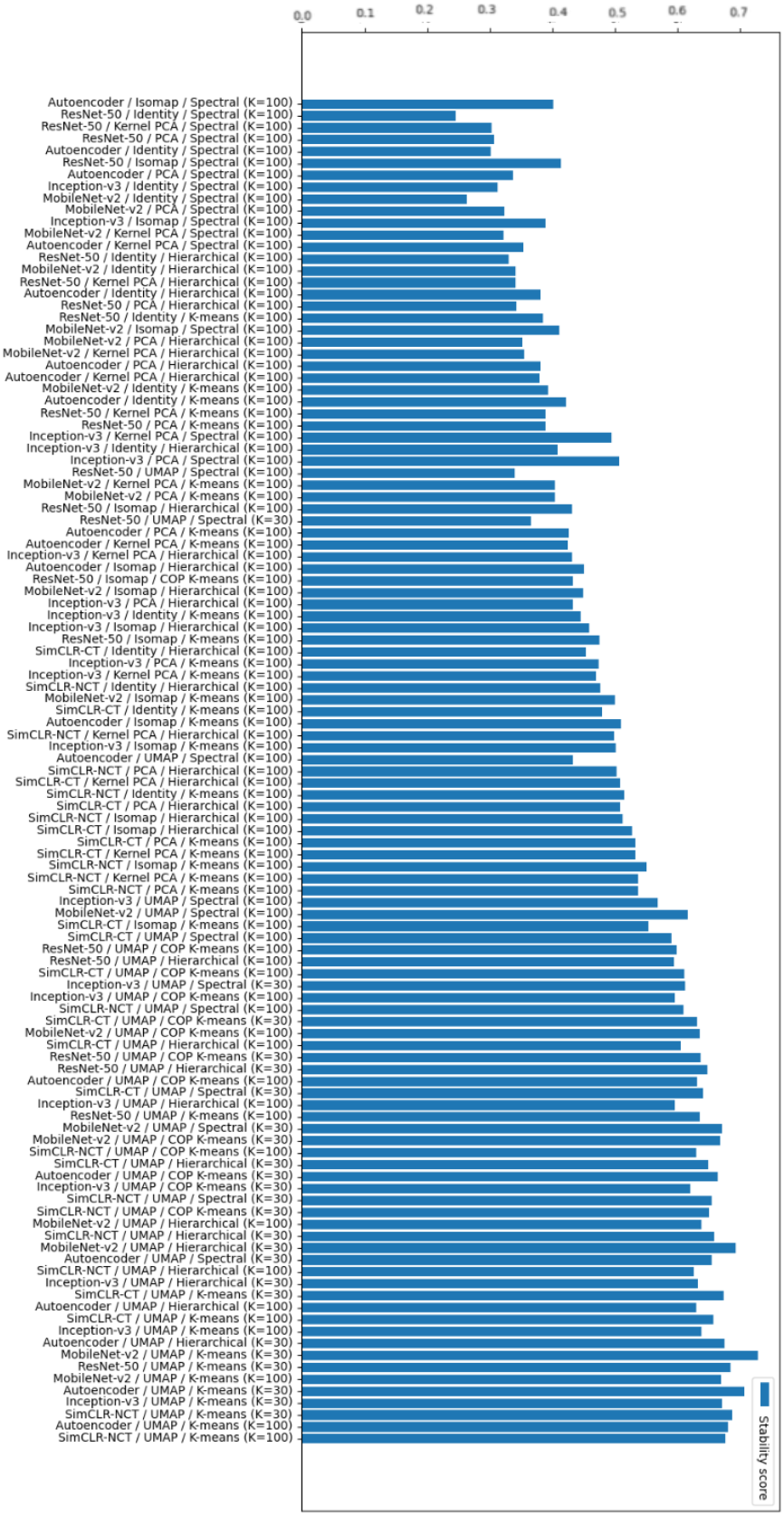


FIGURE 5.1: Ranking of clustering solutions according to the Silhouette score

Feature extractor	Number of results	Majority threshold	Minimum cluster size	Loss	Number of clusters	Number of outliers
MobileNet-v2	4	0,5	46	181	36	73
ResNet-50	6	0,666	42	170	44	378
Inception-v3	5	0,6	46	112	32	153
SimCLR-CT	5	0,6	30	135	36	75
SimCLR-NCT	4	0,5	46	188	37	20
Autoencoder	4	0,5	34	165	36	7

TABLE 5.1: Chosen sets of parameters minimizing the computed loss for each feature extractor.

Homogeneity analysis

In Figure 5.2, we can visualize random samples from each of the computed clusters, each row corresponding to a different cluster. We can first notice the overall visual homogeneity among samples on a same row. This homogeneity can be expressed along different image characteristics.

- For instance, clusters 0, 6 and 14 all present bright auroras located at the bottom of the image, near the south direction.
- The image texture is also taken into account, as we can see with cluster 29, where most samples have the same granular texture, which could correspond to 'patchy' auroras.
- In cluster 7, all samples contain bright auroral structure on a dark background, which shows that contrast can be a determining feature.
- color also plays a role in the classification, as clusters 3, 18, 25 and 34 seem to contain mostly reddish hues in addition to the common green auroral color. This corresponds to either a red auroral emission or the crepuscular sunlight which can appear when the sun is closer to the horizon.
- Some clusters are based more on the image structure, such as cluster 9 which is mainly composed of auroras having multiple parallel lines, with different orientations.
- The presence of very bright white spots caused by the Moon creates a few separate clusters (clusters 11, 15, 28 and 30) and is a predominant feature compared to other image characteristics. This is an important bias which masks the underlying auroral structure information.

It is important to note here that this homogeneity along classes is not systematically due to the chronological proximity of samples from a same cluster. This problem can still be observed in cluster 10, where we can see a chunk of snow on the top of the images biasing the representation.

Structure analysis

If we look at the reduced features data set with their assigned label along with the reference samples, we can extract information about the overall structure of the representation (Figure 5.3). At first, we can see the computed clusters are not always convex and have diverse sizes, which is a result of the clustering fusion process. We can see that there are two main structures: an upward branch and a wider base, extending to the right.

The upward branch, composed of clusters 10, 11, 15, 25, 28, 30 and 34 is mainly due to the presence of the moon. The further up the branch a sample is, the more the Moon is

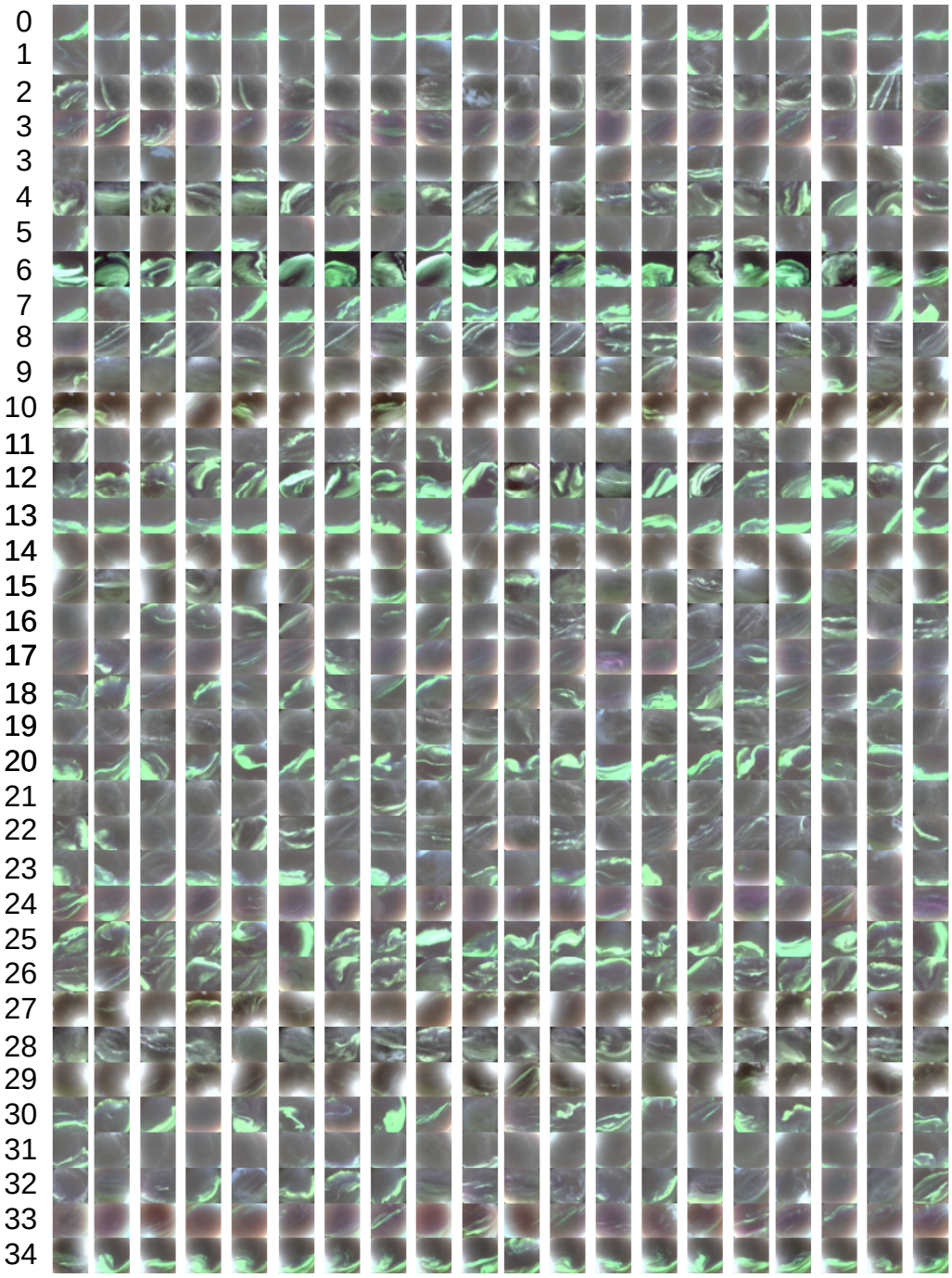


FIGURE 5.2: Display of 20 random samples from each cluster computed by the fusion of clustering results based on the SimCLR-NCT feature extractor.

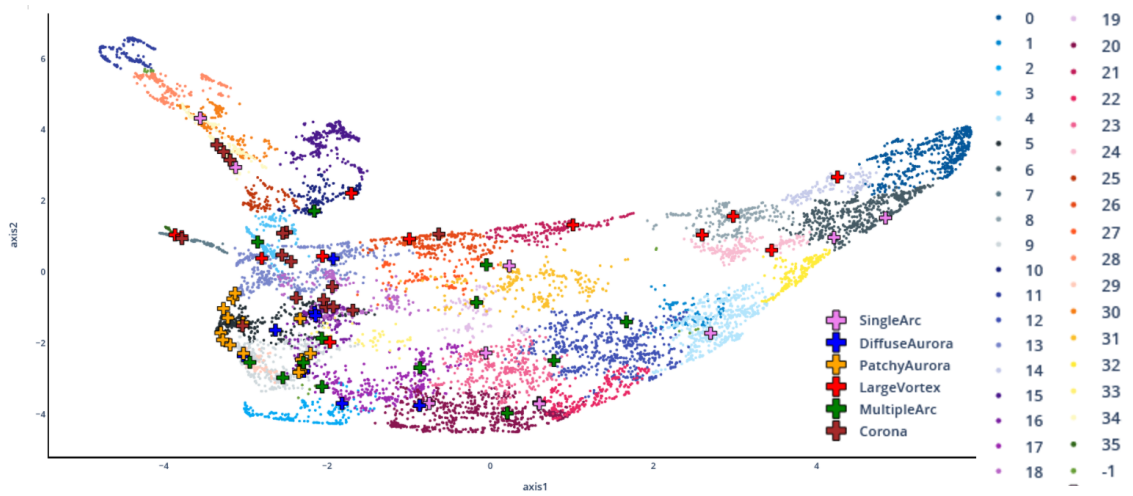


FIGURE 5.3: 2-dimensional representation of the feature data from SimCLR-NCT, with their computed labels and the reference samples.

present in the image. However, for cluster 34 and at the base of the branch with clusters 3 and 18, the bias is instead caused by the crepuscular sunlight. Among those clusters, we find 'Corona' and 'Single arc' auroras, which are indeed likely to occur when the sun is not far from the horizon. Indeed, these two types of aurora occur often around noon, which is when the sky is most affected by sunlight [50].

The small leftmost branch, composed of clusters 35 and 7 is mainly composed of contrasted auroras, on a darker background. These can correspond to vortex-like structures.

The bottom left region, composed of clusters 2, 5, 9, 16 and 29 is characterized by relatively dim images with complex structures or cloud-like auroras. Almost all 'Patchy' auroras end up in this area. Some 'Diffuse' and 'MultipleArc' are also present in this region of the feature space.

Clusters 1, 4, 12, 22 which are in the bottom right region correspond to images where aurora is almost absent from the image, either because the aurora is very dim, or because it is very low on the horizon and is thus cropped at the preprocessing step. Most quiet southward 'SingleArc' auroras end up in this region.

When transitioning to the area which is more on the upper right, with clusters 32, 0, 6 and 14, auroras become more predominant in the image, especially in the south side (bottom of the image).

If we follow towards the left along the top side, the auroral structure extend more toward the north side as well as the east and west. These are more active auroras with more irregular shapes. We find most of the 'Vortex' auroras in these regions.

Almost all 'Corona' samples which have not been biased by the moon are packed in cluster 18, which is around the center of the data points cloud. Then depending on the importance of the bias, they end up in either the cluster 3 or 34.

'Diffuse' auroras are mostly close to 'Patchy' and 'Corona' classes and are thus often confused. Indeed, we can see that these three classes are visually similar, and there are sometimes two types of auroras in the same image.

The 'Vortex' class is widely spread across the the data points cloud. Indeed, because this class is very diverse in terms of shape and texture, it is can be expressed across many different regions of the feature space.

'MultipleArc' auroras are also spread across many clusters, and are often confused with 'Patchy' or 'SingleArc' classes. Because the structure is not clearly visually defined and is often dim, it can be easily confused with other types of aurora.

When they are not affected by the Moon or Sun bias, 'SingleArc' auroras are spread along the bottom right border of the data point cloud.

5.2.2 Solution from other extractors

The representations computed from the other feature extractors seem less balanced and well-defined than the SimCLR-NCT representation (see Annex B), however some can still be interesting to consider depending on the type of aurora and the type of analysis to consider.

For instance, the representation from Inception manages to pack together most of the 'SingleArc' auroras, however the bias introduced by the moon distorts strongly the whole representation, which affects the clustering process.

The Resnet representation presents this same issue, without any meaningful grouping of the different auroral types.

In the Mobilenet feature space, The 'Patchy' and 'Vortex' auroras are well packed while other types are spread. However in this case neighboring samples are often originating from the same auroral events, because they are chronologically close. This is a form of data overfitting which ought to be avoided.

The Autoencoder representation has the particularity of being sensitive to the location of bright spots. For instance, there are distinct clusters corresponding to the different locations of the Moon in the image.

The SimCLR-CT provides a good packing of the 'Patchy' and the 'Vortex' classes, while other auroral types are more spread. We can see no apparent bias from the Moon, which means it is not significantly impacting the representation.

6 Conclusion

In this project, our goal was to automatically identify groups of similar auroral structures solely based on all-sky color images from the Kjell Henriksen Observatory in Svalbard. On a data set of only images containing auroras in clear skies, we have used features extraction methods adapted to this context, based on the scientific literature. On each of the resulting representation of the dataset, we applied dimension reduction methods allowing to keep the most important features for explaining the variations among the auroral images. In particular, the UMAP method provided the best results for the next steps. We then applied a variety of clustering methods including a constrained algorithm which uses a small number of manual labels. Because each of them presented some weaknesses and because the number of clusters is hard to define, we combined the best clustering results according to internal validity scores and visual inspection, using a majority voting principle.

This allowed to have a robust grouping for each representation of the data set. The results are especially meaningful for the representation extracted by the SimCLR-NCT, where the different groups can be interpreted with perceptual and physical criteria. The main issue which remains here is the bias introduced by the presence of the Moon on a number of images, which can in some cases distort the computed representation and mask the underlying similarities of the auroral structures.

In order to avoid this issue, future work should include a processing step excluding all images where the Moon is too high in the sky and/or when it is near the full Moon. This could be done by comparing the time of acquisition of each image with a lunar calendar. The issue of selecting the right number of clusters could also be mitigated by using clustering methods involving no such parameter, with for instance the DBSCAN method. Using a larger number of labelled data could also beneficial not only to analyse the results, but also to better constrain the clustering methods such as COP K-Means. Other constrained clustering methods could be used, such as the Constrained Hierarchical clustering algorithm. Because the majority voting algorithm involves excluding some of the data samples, we could use other ensemble clustering methods to avoid this issue. Finally, the feature extraction methods ought to be explored more in detail, as it is the most important step for extracting meaningful information based on the raw images.

The results found during this project will allow to better understand the variations in auroral morphology as well as to provide unbiased automatic groupings of the auroral images which could be used in statistical studies of the occurrence of the diverse auroral forms. Studying the temporal evolution of the types of aurora and confronting them to the physical description of the auroral events could provide insightful information for understanding this phenomenon.

A Rankings of clustering results

In this annex you can find the ranking of the clustering solutions plotted according to the Calinski-Harabasz score (Figure A.1), the Davies-Bouldin score (Figure A.2) and the Stability score (Figure A.3).

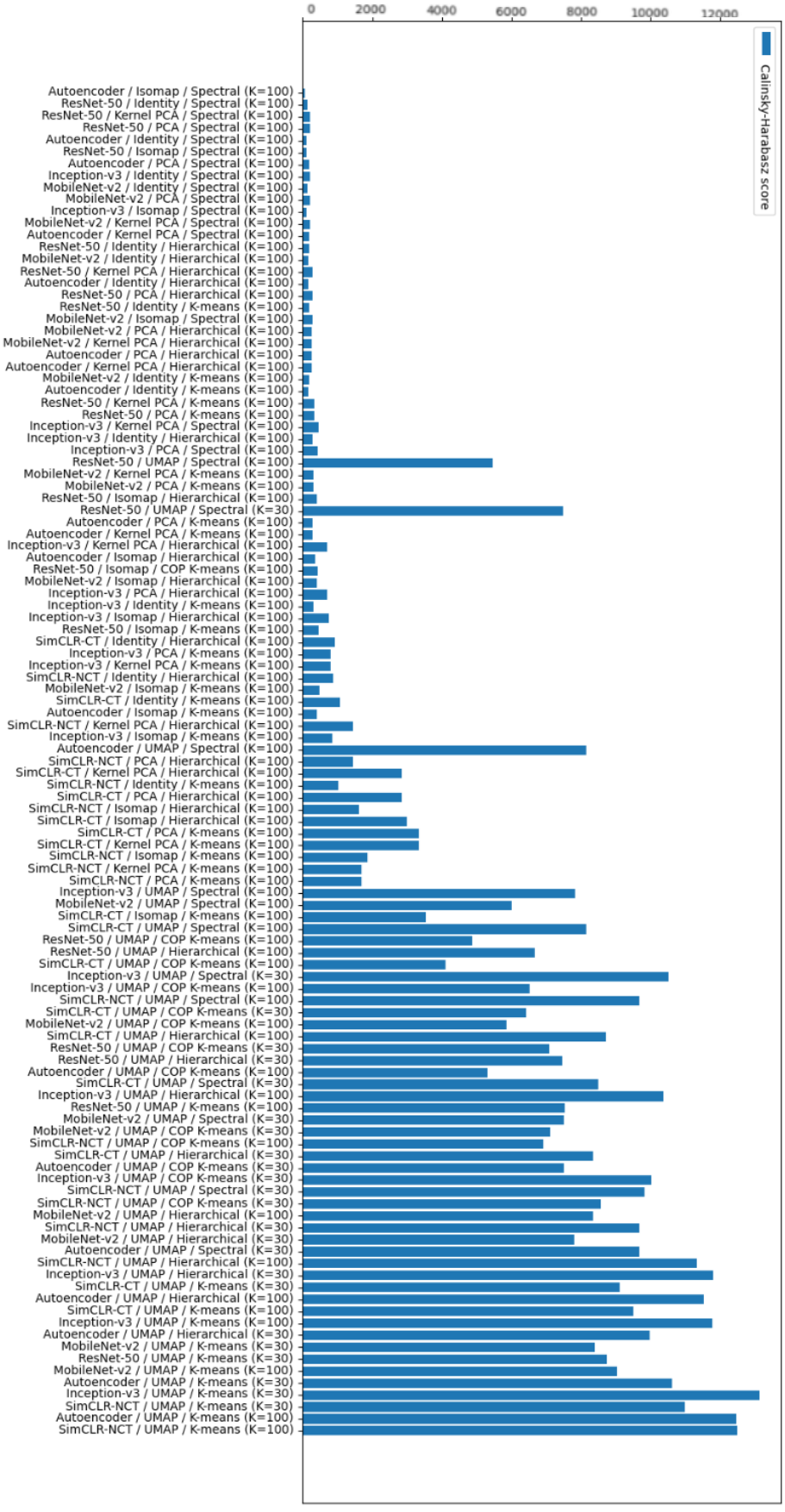


FIGURE A.1: Ranking of clustering solutions according to the Calinski-Harabasz score

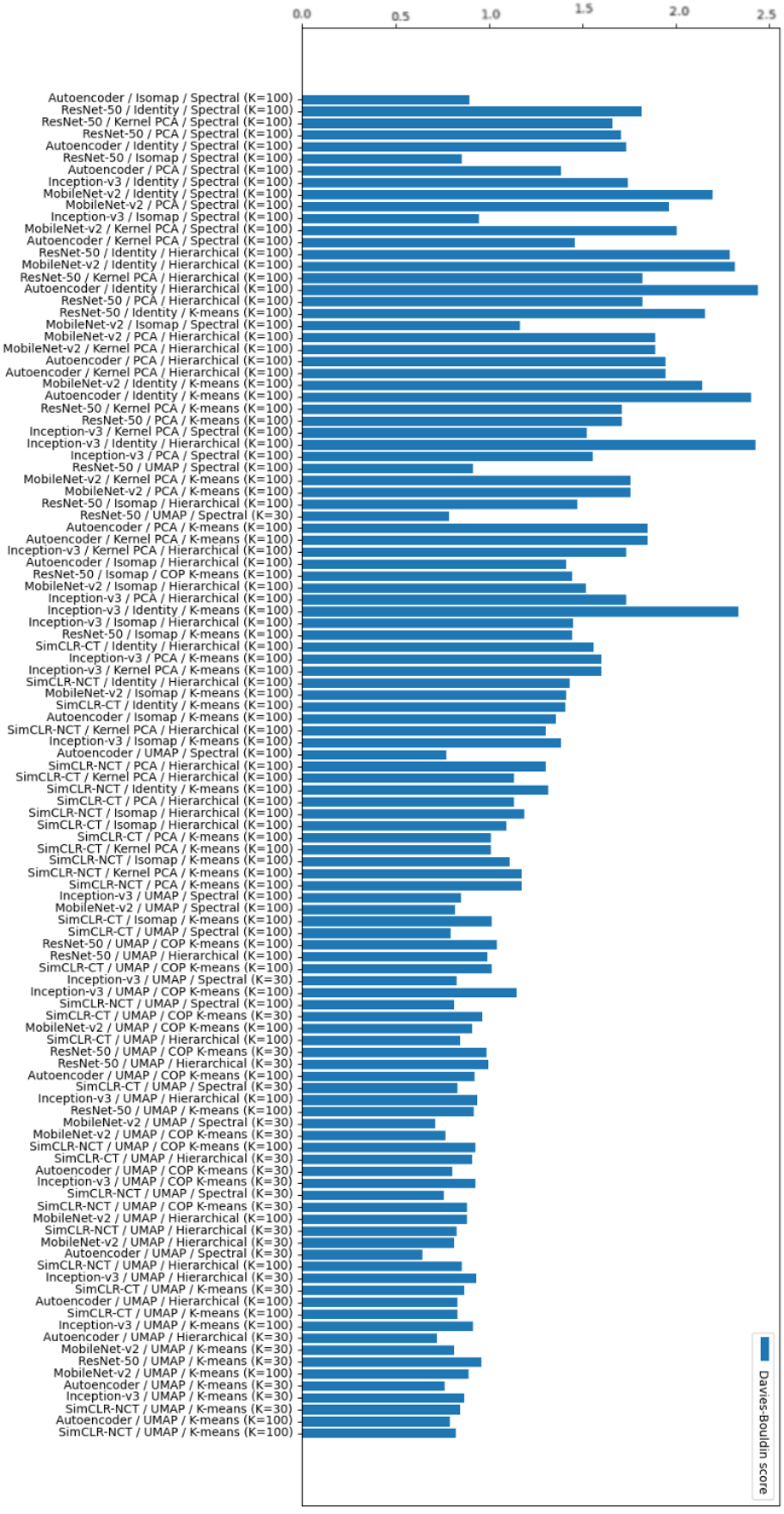


FIGURE A.2: Ranking of clustering solutions according to the Davies-Bouldin score

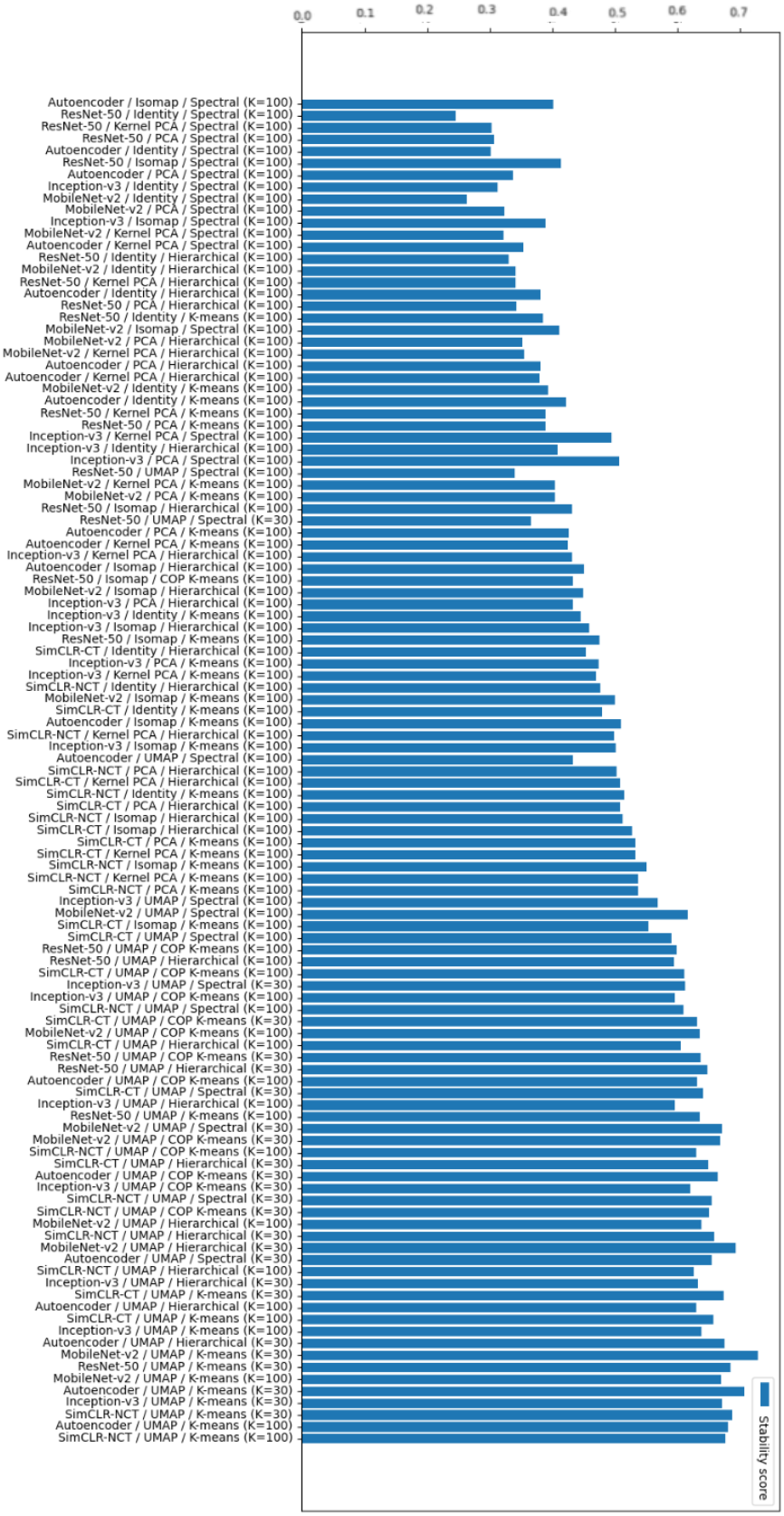


FIGURE A.3: Ranking of clustering solutions according to the Stability score

B 2-dimensional representations of feature data

The following figures show the 2-dimensional representations of the feature data extracted using the 5 computed methods, other than SimCLR-NCT.

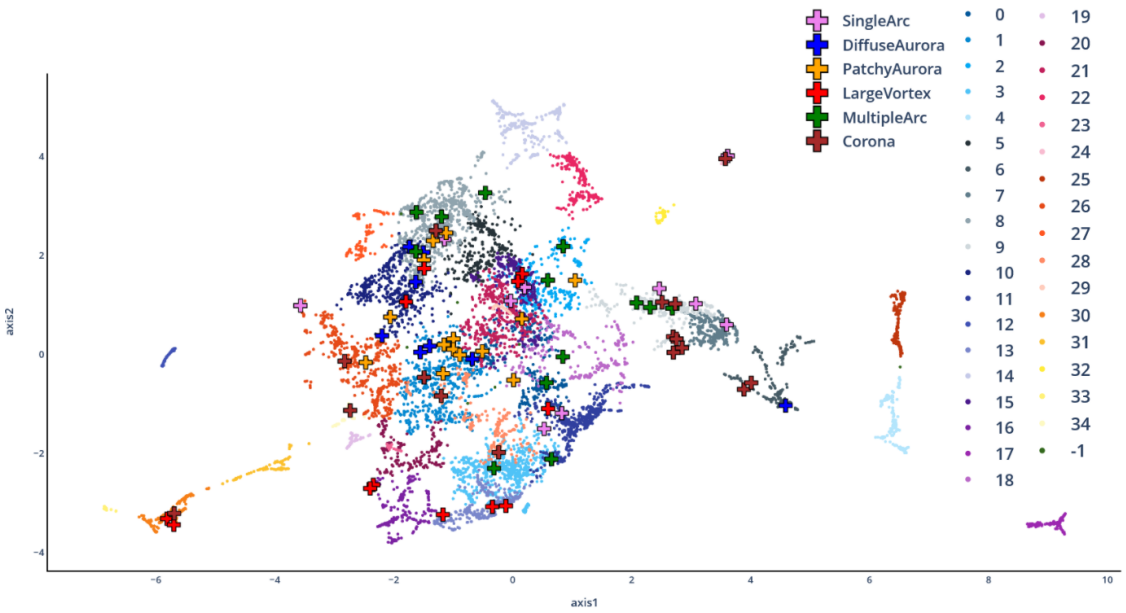


FIGURE B.1: 2-dimensional representation of the feature data from the autoencoder, with their computed labels and the reference samples.

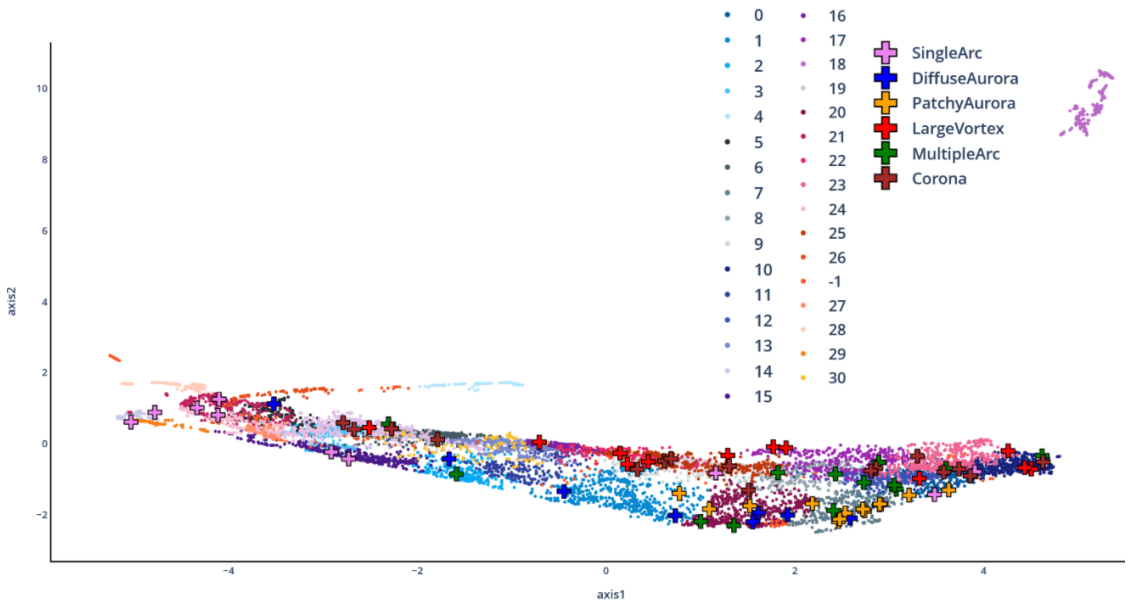


FIGURE B.2: 2-dimensional representation of the feature data from Inception-v3, with their computed labels and the reference samples.

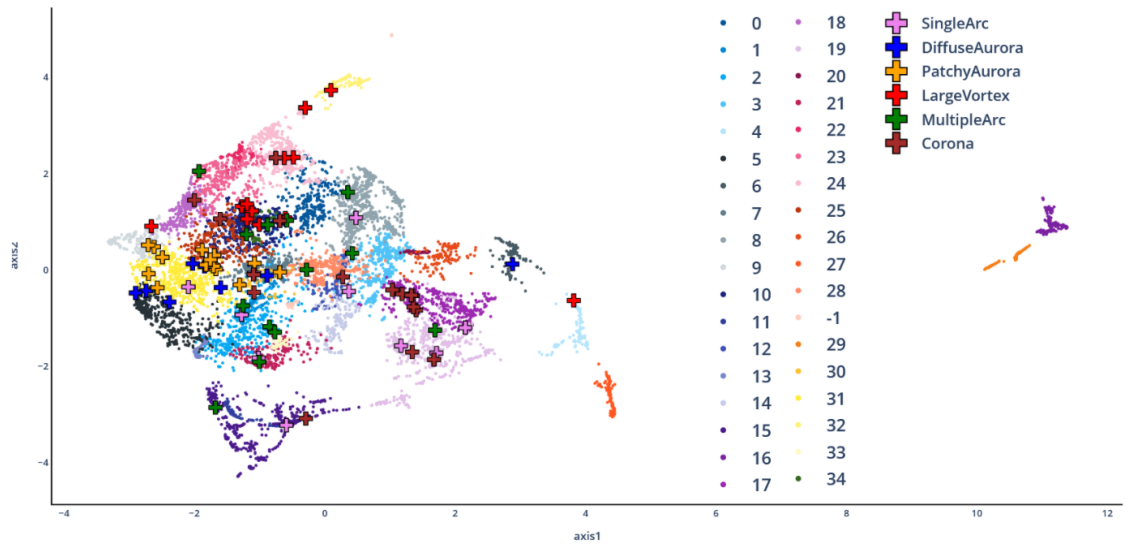


FIGURE B.3: 2-dimensional representation of the feature data from Mobilenet-v2, with their computed labels and the reference samples.

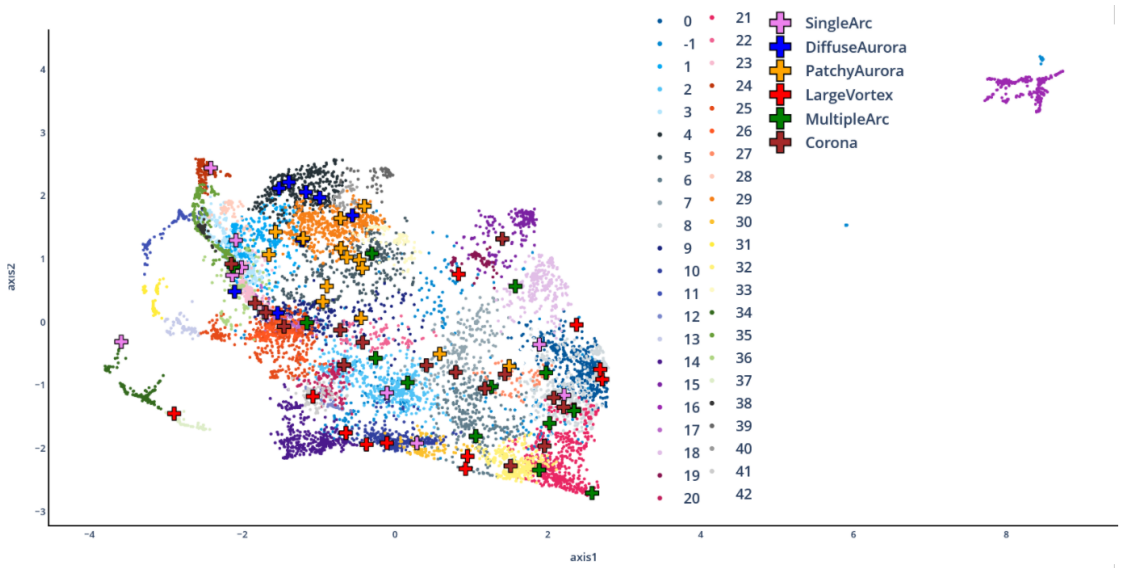


FIGURE B.4: 2-dimensional representation of the feature data from ResNet-50, with their computed labels and the reference samples.

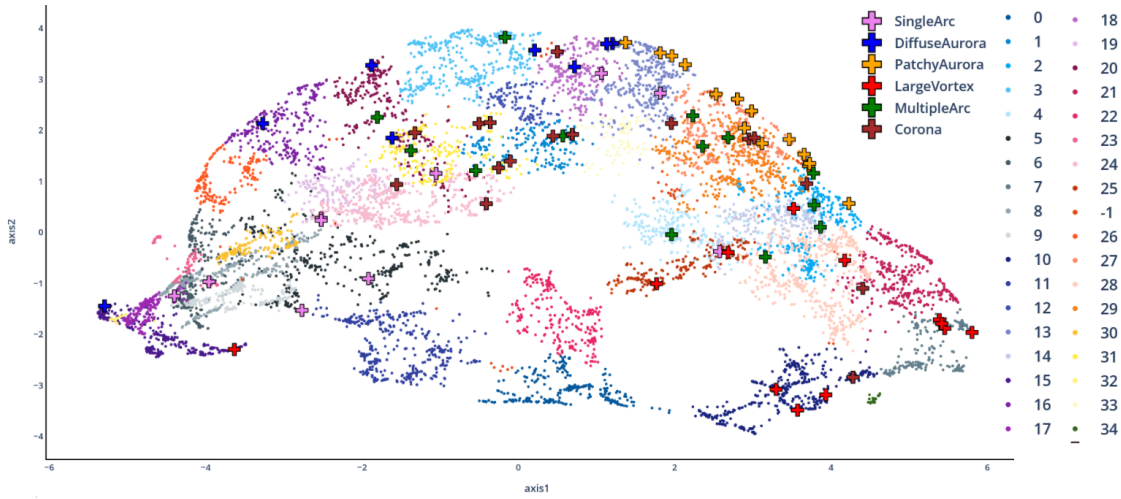


FIGURE B.5: 2-dimensional representation of the feature data from SimCLR-CT, with their computed labels and the reference samples.

C Gantt diagrams of the project

Week number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
READING	Reading of scientific literature of the state-of-the-art								Learn more about established auroral morphologies														
TECHNICAL TASKS	Exploration of the data			Implementations of simple end-to-end solutions and definition of the generic processing chain			Implementation of different solutions into the processing chain, and automatic optimization of hyperparameters					Confront the results of the best solutions on new testing data to the scientific interpretation of auroral morphology				Presentation, analysis and interpretation of experimental results on auroral morphology classification						Cleaning and testing of the code for future utilisation	
DETAILS (in short term)	Statistical study of the dataset		Implementation of the preprocessing step on visual and statistcal criteria		Definition of the different combinations to test and performance criteria			Definition of optimization method and parameter space															
WRITING	Writing of preliminary report				Redaction of the state-of-the-art				Redaction of the project report													Report proofreading	
REPORTS		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report	

Week number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
READING	Reading of scientific literature of the state-of-the-art and auroral physics																						
TECHNICAL TASKS	Exploration of the data			Implementations of simple end-to-end solutions and definition of the generic processing chain			Implementation of different processing blocks (feature extractors, reducers, clustering methods) into the processing chain					Experimenting and comparing different solutions using performance scores. Adding some new blocks to compensate issues found.				Final strategy definition	Application of final strategy				Saving final results	Cleaning and testing of the code for future utilisation	
DETAILS	Statistical study of the dataset		Implementation of the preprocessing step on visual and statistcal criteria		Re-organization of first codes		Implementation of intermediate results automated saving system			Testing and debugging all blocks and saving system			Implementing constrained and ensemble clustering methods			Launching all blocks until clustering step and saving all intermediate results		Launching all clustering methods		Choose and analyse best results			
												Experimenting on time-consuming features extractors (CNNs)						Implementation of clustering results selection and fusion					
WRITING	Writing of preliminary report				Redaction of the state-of-the-art and general explanations											Redaction of the methodology and description of the results						Report proofreading	
REPORTS		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report		Bi-monthly report	

TABLE C.1: Gantt diagrams of the project (upper table was made 4 weeks after the beginning of the project and lower table was the actual schedule of the project)

Bibliography

[1] "Kjell henriksen observatory (KHO)." (), [Online]. Available: http://kho.unis.no/ (visited on 08/19/2022).

[2] J. A. Eddy, *The Sun, the Earth, and Near-earth Space: A Guide to the Sun-earth System*. Government Printing Office, 2009, 316 pp., Google-Books-ID: _gTD1eXxlTkC, ISBN: 978-0-16-083808-8.

[3] S. I. Akasofu, "The development of the auroral substorm," *Planetary and Space Science*, vol. 12, no. 4, pp. 273–282, Apr. 1, 1964, ISSN: 0032-0633. DOI: 10.1016/0032-0633(64)90151-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0032063364901515 (visited on 05/11/2022).

[4] Q. Yang, C. Liu, and J. Liang, "Unsupervised automatic classification of all-sky auroral images using deep clustering technology," *Earth Science Informatics*, vol. 14, no. 3, pp. 1327–1337, Sep. 1, 2021, ISSN: 1865-0481. DOI: 10.1007/s12145-021-00634-1. [Online]. Available: https://doi.org/10.1007/s12145-021-00634-1 (visited on 04/04/2022).

[5] A. Kvammen, K. Wickstrøm, D. McKay, and N. Partamies, "Auroral image classification with deep neural networks," *Journal of Geophysical Research: Space Physics*, 2020. DOI: 10.1029/2020JA027808.

[6] I. A. o. G. a. A. A. Committee, *International auroral atlas: published for the International Union of Geodesy and Geophysics*. Edinburgh: University Press; [North American agent: Aldine Pub. Co., Chicago], 1963, 17 pp., Open Library ID: OL5903169M.

[7] P. Sado, L. B. N. Clausen, W. J. Miloch, and H. Nickisch, "Transfer learning aurora image classification and magnetic disturbance evaluation," *Journal of Geophysical Research: Space Physics*, vol. 127, no. 1, e2021JA029683, 2022, ISSN: 2169-9402. DOI: 10.1029/2021JA029683. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2021JA029683 (visited on 08/19/2022).

[8] L. B. N. Clausen and H. Nickisch, "Automatic classification of auroral images from the oslo auroral THEMIS (OATH) data set using machine learning," *Journal of Geophysical Research: Space Physics*, vol. 123, no. 7, pp. 5640–5647, 2018, ISSN: 2169-9402. DOI: 10.1029/2018JA025274. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2018JA025274 (visited on 08/19/2022).

[9] M. T. Syrjäsuo and E. F. Donovan, "Diurnal auroral occurrence statistics obtained via machine vision," *Annales Geophysicae*, vol. 22, no. 4, pp. 1103–1113, Apr. 2, 2004, Publisher: Copernicus GmbH, ISSN: 0992-7689. DOI: 10.5194/angeo-22-1103-2004. [Online]. Available: https://angeo.copernicus.org/articles/22/1103/2004/ (visited on 03/15/2022).

[10] J. W. Johnson, S. Hari, D. Hampton, H. K. Connor, and A. Keesee, "A contrastive learning approach to auroral identification and classification," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2021, pp. 772–777. DOI: 10.1109/ICMLA52953.2021.00128.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and*

Pattern Recognition, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[12] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New Directions in Statistical Physics: Econophysics, Bioinformatics, and Pattern Recognition*, L. T. Wille, Ed., Berlin, Heidelberg: Springer, 2004, pp. 273–309, ISBN: 978-3-662-08968-2. DOI: 10.1007/978-3-662-08968-2_16. [Online]. Available: https://doi.org/10.1007/978-3-662-08968-2_16 (visited on 08/19/2022).

[13] M. Syrjäsuo, "Automatic classification of auroral images in substorm studies," *Proc. 8th ICS*, pp. 309–313, Jan. 1, 2007.

[14] X. Qin and Y.-H. Yang, "Basic gray level aura matrices: Theory and its application to texture synthesis," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, ISSN: 2380-7504, vol. 1, Oct. 2005, 128–135 Vol. 1. DOI: 10.1109/ICCV.2005.43.

[15] J. Rao, N. Partamies, O. Amariutei, M. Syrjäsuo, and K. E. A. van de Sande, "Automatic auroral detection in color all-sky camera images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 12, pp. 4717–4725, Dec. 2014, Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2014.2321433.

[16] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Sep. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[17] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, Nov. 1, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[19] S. Basodi, C. Ji, H. Zhang, and Y. Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 196–207, Sep. 2020, Conference Name: Big Data Mining and Analytics, ISSN: 2096-0654. DOI: 10.26599/BDMA.2020.9020004.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[21] A. G. Howard, M. Zhu, B. Chen, *et al.*, *MobileNets: Efficient convolutional neural networks for mobile vision applications*, Number: arXiv:1704.04861, Apr. 16, 2017. DOI: 10.48550/arXiv.1704.04861. arXiv: 1704.04861[cs]. [Online]. Available: http://arxiv.org/abs/1704.04861 (visited on 08/19/2022).

[22] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, JMLR.org, Jul. 13, 2020, pp. 1597–1607. (visited on 08/19/2022).

[23] J. Shlens, *A tutorial on principal component analysis*, Number: arXiv:1404.1100, Apr. 3, 2014. DOI: 10.48550/arXiv.1404.1100. arXiv: 1404.1100[cs,stat]. [Online]. Available: http://arxiv.org/abs/1404.1100 (visited on 08/19/2022).

[24] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1, 1998, ISSN: 0899-7667. DOI: 10.1162/089976698300017467. [Online]. Available: https://doi.org/10.1162/089976698300017467 (visited on 08/19/2022).

[25] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 22, 2000, Publisher: American Association for the Advancement of Science. DOI: 10.1126/science.290.5500.2319. [Online]. Available: https://www.science.org/doi/10.1126/science.290.5500.2319 (visited on 08/19/2022).

[26] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, Jun. 1, 1962, ISSN: 0001-0782. DOI: 10.1145/367766.368168. [Online]. Available: https://doi.org/10.1145/367766.368168 (visited on 08/19/2022).

[27] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html (visited on 08/19/2022).

[28] L. McInnes, J. Healy, and J. Melville, *UMAP: Uniform manifold approximation and projection for dimension reduction*, Number: arXiv:1802.03426, Sep. 17, 2020. DOI: 10.48550/arXiv.1802.03426. arXiv: 1802.03426[cs,stat]. [Online]. Available: http://arxiv.org/abs/1802.03426 (visited on 08/19/2022).

[29] G. C. Linderman and S. Steinerberger, "Clustering with t-SNE, provably," *SIAM journal on mathematics of data science*, vol. 1, no. 2, pp. 313–332, 2019, ISSN: 2577-0187. DOI: 10.1137/18m1216134.

[30] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-SNE effectively," *Distill*, vol. 1, no. 10, e2, Oct. 13, 2016, ISSN: 2476-0757. DOI: 10.23915/distill.00002. [Online]. Available: http://distill.pub/2016/misread-tsne (visited on 08/19/2022).

[31] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982, Conference Name: IEEE Transactions on Information Theory, ISSN: 1557-9654. DOI: 10.1109/TIT.1982.1056489.

[32] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 1, 2007, ISSN: 1573-1375. DOI: 10.1007/s11222-007-9033-z. [Online]. Available: https://doi.org/10.1007/s11222-007-9033-z (visited on 08/19/2022).

[33] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger, "SpectralNet: Spectral clustering using deep neural networks," presented at the 6th International Conference on Learning Representations, ICLR 2018, 2018. [Online]. Available: https://cris.tau.ac.il/en/publications/spectralnet-spectral-clustering-using-deep-neural-networks (visited on 08/19/2022).

[34] F. Nielsen, "Hierarchical clustering," in Feb. 4, 2016, pp. 195–211, ISBN: 978-3-319-21902-8. DOI: 10.1007/978-3-319-21903-5_8.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, Aug. 2, 1996, pp. 226–231. (visited on 08/19/2022).

[36] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, no. 3, 19:1–19:21, Jul. 31, 2017, ISSN: 0362-5915. DOI: 10.1145/3068335. [Online]. Available: https://doi.org/10.1145/3068335 (visited on 08/19/2022).

[37] S. VEGA-PONS and J. RUIZ-SHULCLOPER, "A SURVEY OF CLUSTERING ENSEMBLE ALGORITHMS," *International Journal of Pattern Recognition and Artificial Intelligence*, Nov. 21, 2011, Publisher: World Scientific Publishing Company. DOI: 10.1142/S0218001411008683. [Online]. Available: https://www.worldscientific.com/doi/epdf/10.1142/S0218001411008683 (visited on 04/29/2022).

[38] A. Fred, "Finding consistent clusters in data partitions," in *Multiple Classifier Systems*, J. Kittler and F. Roli, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2001, pp. 309–318, ISBN: 978-3-540-48219-2. DOI: 10.1007/3-540-48219-9_31.

[39] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 28, 2001, pp. 577–584, ISBN: 978-1-55860-778-1. (visited on 08/19/2022).

[40] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1, 1987, ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257 (visited on 08/19/2022).

[41] T. Caliński and J Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, Jan. 1, 1974, ISSN: 0090-3272. DOI: 10.1080/03610927408827101. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/03610927408827101 (visited on 08/19/2022).

[42] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN: 1939-3539. DOI: 10.1109/TPAMI.1979.4766909.

[43] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, Dec. 1, 1971, ISSN: 0162-1459. DOI: 10.1080/01621459.1971.10482356. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356 (visited on 08/19/2022).

[44] S. Ben-David and U. Luxburg, "Relating clustering stability to properties of cluster boundaries.," presented at the Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008), 379-390 (2008), Jan. 1, 2008, pp. 379–390.

[45] A. K. Jain, *Fundamentals of digital image processing*. USA: Prentice-Hall, Inc., 1989, 569 pp., ISBN: 978-0-13-336165-0.

[46] K. Team. "Keras documentation: Keras applications." (), [Online]. Available: https://keras.io/api/applications/ (visited on 08/19/2022).

[47] "UMAP: Uniform manifold approximation and projection for dimension reduction — umap 0.5 documentation." (), [Online]. Available: https://umap-learn.readthedocs.io/en/latest/index.html (visited on 08/19/2022).

[48] "2.3. clustering," scikit-learn. (), [Online]. Available: https://scikit-learn/stable/modules/clustering.html (visited on 08/19/2022).

[49] B. Babaki, *COP-kmeans*, original-date: 2015-10-17T12:30:34Z, Jul. 6, 2022. [Online]. Available: https://github.com/Behrouz-Babaki/COP-Kmeans (visited on 08/19/2022).

[50] Q. Wang, J. Liang, Z.-J. Hu, *et al.*, "Spatial texture based automatic classification of dayside aurora in all-sky images," *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 72, no. 5, pp. 498–508, Apr. 1, 2010, ISSN: 1364-6826. DOI: 10.1016/j.jastp.2010.01.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364682610000441 (visited on 08/19/2022).